68' MICRO JOURNAL

Australia	A $4.75	New Zealand	NZ $ 6.50
Singapore	S $9.45	Hong Kong	H $23.50
Malaysia	M $9.45	Sweden	30:-SEK

$2.95USA

OS-9 Atari Amiga Mac S-50

6800 6809 68008 68000 68010 68020 68030

The Magazine for Motorola CPU Devices *For Over a Decade!*

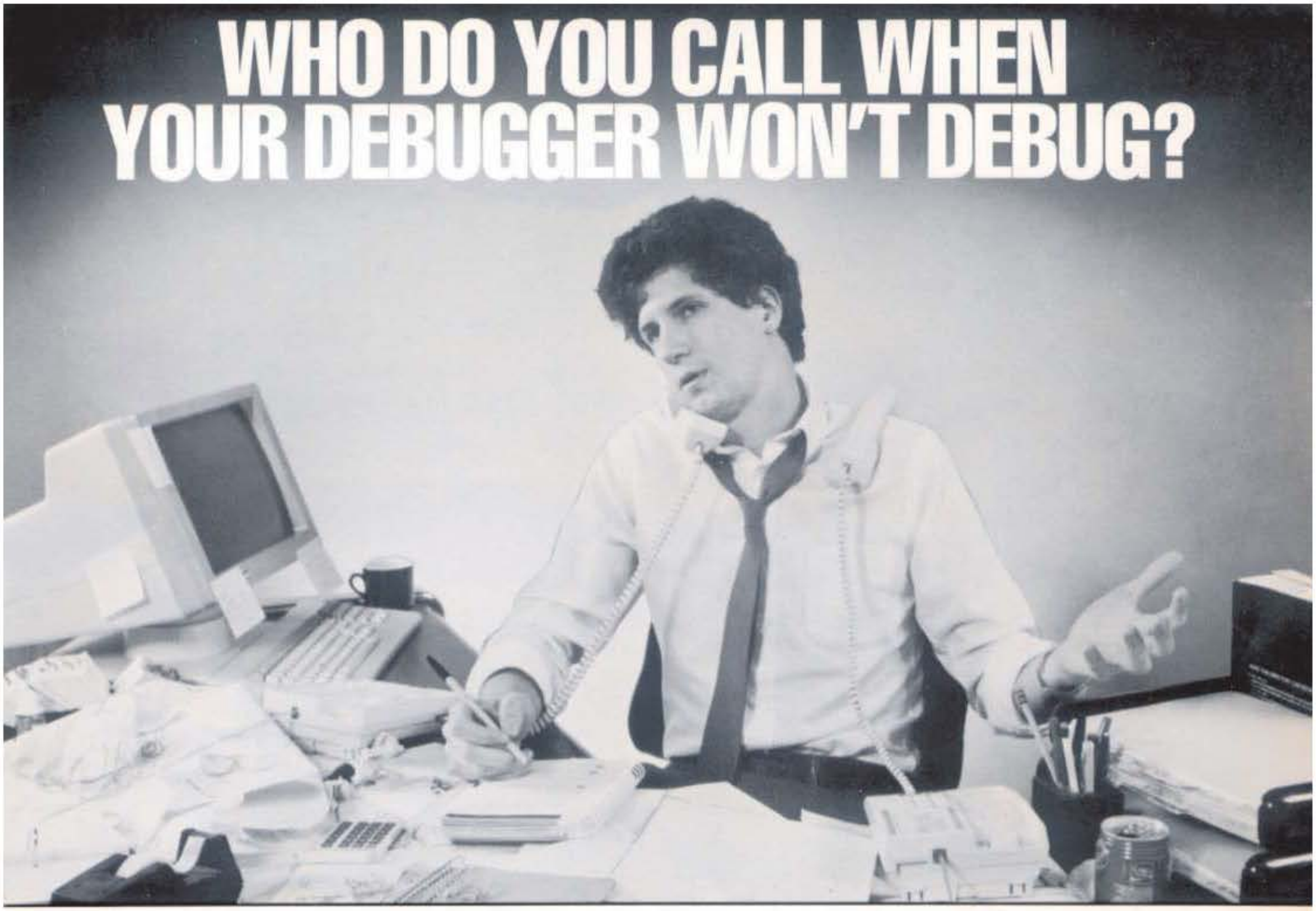
This Issue:

OS-9 SK*DOS Atari Amiga FLEX Macintosh *A User Contributor Journal* And Lots More!

VOLUME X ISSUE XII ● Devoted to the 68XXX User ● December 1988

The Grandfather of "DeskTop Publishing™"

PHOTO CREDIT: NASA



The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

Why Not Try the Microware One-Stop Total Solution?

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Modularity Lets YOU Choose Just What You Need.

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

The OS-9 Success Kit

A Total Integrated Solution for Your Next Project

Development Tools:

C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

Kernel Options:

MMU (Security Protection) Support
Math Coprocessor Support

Languages:

C*
Basic
Pascal
Fortran
Ada**
Assembler*

I/O Options:

SCSI, SASI & SMD Disks
3-, 5-, 8-Inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

*Resident or UNIX versions available
**VAX hosted

Support is Part of the Package.

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT&T.
VAX is a trademark of DEC.

microware® OS-9

Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

Microware Japan Ltd.
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

Christmas Special

Mustang-020 Mustang-08 Benchmarks

	32 bit Integer	Register Long
IBM AT 7300 Xenix Sys 3	[illegible]	[illegible]
AT&T 7300 UNIX PC 68010	7.[illegible]	[illegible]
DEC VAX 11/780 UNIX Berkley 4.2	[illegible]	[illegible]
DEC VAX 11/750 " "	[illegible]1	3.2
68008 OS-9 68K 8 Mhz	[illegible]	[illegible]
68000 OS-9 68K 10 Mhz	6.5	4.0
MUSTANG-08 68008 OS-9 68K 10 Mhz	*9.8*	*6.3*
MUSTANG-020 68020 OS-9 68K 16 Mhz	*2.2*	*0.88*
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz	*1.8*	*1.22*

```
Main()
            {
                register long i;
                for (i=0; i < 999999; ++i);
            }
```

Estimated MIPS - MUSTANG-020 — 4.5 MIPS,

Burst to 8 - 10 MIPS: Motorola Specs

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

CPI
Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020

OS-9

OS-9 Professional Ver *Includes C Compiler	$850.00
Basic09	300.00
C Compiler	500.00
68000 Disassembler (w/source add: $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor + (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor + (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz.

Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232	335.00
Requires 1 or 2 Adaptor Cards below RS232 Adaptor	165.00

Each card supports 4 additional ser. ports
(total of 36 serial ports supported)

60 line Parallel I/O card	398.00

Uses 3 68230 Interface/Timer chips,
6 groups of 8 lines each, separate buffer
direction control for each group.

Prototype Board	75.00

area for both dip and PGA devices & a
pre-wired memory area up to 512K DRAM.

SBC-AN	475.00

Interface between the system and
ARCNET modified token-passing LAN, fiber optics optional - call.

LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

Special for complete MUSTANG-020™ system buyers - Sculptor+
$695.00. SAVE $300.00
Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

The
P
R
O
!

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must.

Only the "PRO" Version
of OS-9 Supported!

This is *HEAVY DUTY*
Country!

Price List:

Mustang-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5"-80 track floppy DS/DD		$269.95
Floppy Cable		$39.95
OS-9 68K Professional Version		$850.00
C Compiler ($500 Value)		N/C
Winchester Cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk Controller		$395.00
Shipping USA UPS		$20.00
UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$375.00
20 Mhz MC68020 Sys		$750.00
Note all 68881 chips work with 20 Mhz Sys		
	Total:	$5299.80

For a limited time we will offer a $400
trade-in on your old 68XXX SBC.
Must be working properly and
complete with all software, cables and
documentation.
Call for more information

SAVE UP TO
$1500

Christmas Special
Complete Systems

25 Mbyte HD ~~$4299.80~~ $3799.80
85 Mbyte HD ~~$5748.80~~ $4948.80

Note: Only Professional OS-9 Now Available (68020 Version)
Includes ($500) C Compiler - 68020 & 68881 Supported -
For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

Data-Comp Division
CPI
A Decade of Quality Service
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Cheryl Hodge

Contributing & Associate Editors

Ron Anderson
Ron Voigts
Doug Lurie
Ed Law
Dr. E.M. "Bud" Pass
Art Weller
Dr. Theo Elbert
& Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.
"Over a Decade of Service"
"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343
Phone (615) 842-4600 Telex 510 600-6630

68 Micro Journal is the *original*" DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN. and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Leuers of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

service
For Over a Decade!
1,200 Titles +

A Division of
CPI

S.E.
Media
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU!** **All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO ——— PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

C

The C Programmers Reference Source. Always Right On Target!

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter begins the presentation of a binary file editor and discusses byte-ordering considerations affecting it and similar programs.

BYTE ORDERING

One of the portability problems which must be addressed by the writer, maintainer, or porter of a program such as the one described below which deals with the internal representation of numbers and strings is the problem of byte-ordering.

When dealing with 8-bit characters and 16-bit words, there are only two possible orderings of characters within the word. Thus, if a word logically contains the character string "ab", the characters will actually be ordered in memory as "a", then "b", or as "b", then "a".

However, when dealing with 8-bit characters and 32-bit words, there are 4 factorial = twenty-four possible orderings of characters within the word, although not all are found in current machine/compiler combinations. Thus, if a longword logically contains the character string "abcd", the characters will actually be ordered in memory in one of the following forms:

abcd (680x, 680x0)
abdc
acbd
acdb
adbc
adcb
bacd
badc
bcad
bcda
bdac
bdca
cabd
cadb
cbad
cbda
cdab
cdba
dabc
dacb
dbac
dbca
dcab
dcba (8086, 8088, 80x86, VAX)

Luckily, this difference is normally obscured by the particular version of the C compiler being used. Only when attempting to move data or programs from one machine or compiler to another, to convert from one data representation to another, or access the hardware directly will this difference usually become aParent.

In the case of the program described below, this difference is important because the program is intended to be portable and because it must deal with the individual characters in a word or longword. Luckily, the ZAP program need only be concerned with word ordering.

THE ZAP PROGRAM

Following is the description of a program named zap.

Zap provides a binary file inspection and patching facility. Thus, it allows the in-place modification of existing files, with the restriction that characters may not be inserted or deleted.

It was written by Johan Vromans at Multihouse Research, Gouda, the Netherlands. He placed it into the public domain.

Features of zap include the following:

- looking at the file by byte, word or longword,
- displaying contents in octal, hex, decimal and ascii,
- searching for bytes/words/longwords,
- verifying changes,
- buffering update with optional checksum.

Zap is portable, mainly because it does not use system-dependent constructs. It does not use shell escapes, tty status modifications, etc.

The only system dependency important to zap is the details of byte ordering.

It regards the file as a sequence of bytes, words (2 bytes) or longwords (4 bytes). Changes are buffered, and only aPlied to the file upon normal completion. Only real changes are considered a modification, e.g. a change of 1 to 1 is a no-operation, and a change of 1 to 2 and then again to 1 discards the modification.

Input may be redirected from a file for batch-mode patching.

Zap regards locations in a file to be an offset to a base. After invokation, zap asks for the base value. When end-of-file is detected, the program terminates. After the base value has been entered, zap asks for the offset value. End-of-file makes it go back to the "Base" prompt. After the offset value is entered, zap displays the offset, base and contents of the current location, and waits for commands to execute. Typing end-of-file to the command prompt makes zap go back to the "Offset" question.

Zap operates in one of three modes: byte, word or longword. Words and longwords need not be aligned. The current mode is identified by a "\" for byte mode, "/" for word mode, and "l" for longword mode.

The contents of a location are displayed in one of four formats: octal, decimal, hexadecimal or ascii. See the commands how to change this format. In ascii format, some interpretation is made to show special control characters.

An example of zap's output follows.

```
Base ? 0100
Offset ? 0200
Base Offset Value New
000100 000200\ 0130 /
000100 000200/ 054117
000100 000202/ 045200
000100 000204/ 063004 ^Z
Offset ? ^Z
Base ? ^Z
```

Valid zap editing commands are described below. Each command is terminated by a new-line character.

Changing mode:

\ change to byte mode.

/ change to word (2-byte) mode.

l change to longword (4-byte) mode.

Changing display format

;a change display format to ascii. In ascii format, the base and offset values are displayed in octal.

;d change display format to decimal.

;o change display fonnat to octal.

;x change display format to hexadecimal.

Moving around

(new-line) advances to the next location.

^ backs up to the previous location.

>nnn moves offset to the specified location. If nnn is omited, the contents of the current location are used. The current location is saved in the location table. Up to 256 saved locations can be restored in a last-in first-out manner.

< moves back to the most recently saved location.

Modifying contents

nnn - sets the contents of the current location to nnn. This can be an octal, decimal or a hexadecimal number in the form nnn (decimal), 0nnn (octal), 0xnnn (hex). The offset is advanced to the next location.

nnn^ - sets the contents of the current location to nnn as described above. The offset is backed up to the previous location.

;axyz - changes display format to ascii, and stores the ascii character string xyz starting at the current location. The current offset is advanced to the location after the string.

Miscellaneous commands

;v - prints a list of pending modifications.

;s - asks for a search value and boundaries, and then searches for the specfied value. The locations where it is found are printed, and also stored in the location table. The search aPlies to the file contents only. Pending modifications are ignored during the search. The argument to the search can be suPlied numerically (decimal, octal or hex), or in the format ;apqr which causes it to be interpreted as an ascii search argument. The number of characters allowed depends on the current mode of operation: 1 for byte mode, 2 for word mode, and 4 for longword mode. A search can be interrupted using the terminal interrupt signal. Note that the current mode controls the search. If zap is in byte mode, the search is for a byte, and so on. When searching for words or longwords, word boundaries are ignored. During the search, a "." is displayed for each 1024 bytes processed. This can be suPressed with the -s command line option.

^Y (caret-uPercase-Y or control/Y) - terminates the zap loop without asking for new offset/ base values.

^Z (caret-uPercase-Z) - signifies end-of-file.

Program calling sequence

zap [-options] file-name

The following options may be given (in any order) before the file-name argument:

-c calculates a 16-bit checksum involving all modifications. The order in which the modifications are made is not important. Zap requests a checksum value to be entered upon completion, and requires this value to match the checksum. If they differ, no modifications are made.

-d calculates the checksum and prints its value upon completion.

-r accesses the file for inspection only. This is the default.

-s works silently. No prompts and remarks are displayed. This can be used for batch-like processing, if input has been re-directed from a file.

-v suPlies informational messages.

-w accesses the file in a mode which allows modification.

DIAGNOSTICS

no write access: a modification is made, and the -w option was not suPlied. This message is showed only once.

no modifications made: the file has not been modified, either because the -w option was missing, or the requested checksum did not match. This situation is considered an error.

no modifications requested: the file was accessed using the -w option, but no changes were pending. This is an informational message.

input error: an invalid input format was suPlied to a numeric prompt. The question is repeated.

start > end: the end value for a search exceeded the starting position. The search is not executed.

EOF > end, truncated: the end value for a search exceeded the end-of-file. The end-of-file value is used.

you may recompile with "-DSWAB=X": zap determined that your system swaps bytes (SWAB = 1) or not (SWAB = 0). You may use this in a subsequent compilation.

please recompile with "-DSWAB=X": zap determined that your system swaps bytes (SWAB = 1) or not (SWAB = 0), but the oPosite was specified during compilation. You will have to recompile with the correct value.

EXAMPLE C PROGRAM

Following is this month's example C program; it is the first part of zap, as discussed earlier. The remainder is presented in the next chapter.

```
/*
     zap.c - program to inspect/patch binary files

     Written by Johan Vromans at Multihouse Research,
     Gouda, the Netherlands.
     Copyright 1987 Johan Vromans.
     Distribution free as long as you give
     credit to the original author.
     Military use and explicit resale prohibited.
     Usage of this program is at your own risk.
*/

#include <stdio.h>
#include <ctype.h>
#include <signal.h>

#ifndef TRUE
#  define TRUE  1
#  define FALSE 0
#endif

/* define SWAB=1 for byte swaPing machines */
/* such as intel, vax and pdp-11; */

/* define SWAB=0 for non-swaPing machines */
/* such as most motorola; */

/* if unknown, don't define it - zap will find out */

#ifndef SWAB
```

```
# ifdef vax         /* DEC VAX family */
#  define SWAB 1
# endif
# ifdef pdp11       /* DEC PDP-11 family */
#  define SWAB 1
# endif
# ifdef mc68000     /* Motorola 680x0 family */
#   define SWAB 0
# endif
# ifdef M_I86       /* Intel 86 family */
#   define SWAB 1
# endif
#endif

#ifndef SWAB
int swab = FALSE;     /* use dynamic method */
#else
#  define swab  SWAB  /* leave it to the compiler */
#endif

/* About swaPing -
 *
 *      Representation of data
 *
 *                          swaPing    non-swaPing
 *  type        numeric     character   character
 *  byte        0x61        'a'         'a'
 *  word        0x6162      'ba'        'ab'
 *  longword    0x61626364  'dcba'      'abcd'
 */

#define V_fprintf     (void) fprintf
#define V_printf      (void) printf
#define V_sprintf     (void) sprintf
#define ASCII   3
#define BYTE    1
#define DECIMAL 1
#define HEX     2
#define LWORD   4
#define OCTAL   0
#define WORD    2
#define BYTEVAL(x)    ((x) & 0xff)
#define BUF_INC 512
#define PREV_MAX 256  /* size of previous goto table  */
#define put_byte      enter
#define getbyte(addr) BYTEVAL((cur) ? get_value (addr) : gv_file (addr))

char *calloc ();
char *realloc ();
char *strcpy ();
long lseek ();
#ifdef lint
void clearerr ();
#endif
void exit();
char *my_name    = "zap";    /* identification */
char *usage      = "usage: zap [-cdrsvw] file";
int f_batch;                /* running batch mode */
int f_check;                /* request checksum */
int f_silent;               /* silent */
int f_sum;                  /* print checksum */
int f_verbose;              /* give more info */
int f_write;                /* read-write */
```

```
main (argc, argv)
int argc;
char *argv[];
{
    char *arg_ptr;          /* argument pointer */
    char c;                 /* current option character */
    int file_cnt;           /* number of files processed */

    swabcheck (); /* verify or establish swap mode */
    /* ignore first argument (program name) */
    argc-;
    argv++;
    f_batch = !isatty (0);
    file_cnt = 0;                   /* haven't seen one yet */
    while (argc— > 0)              /* through arguments */
    {
        /* fetch a pointer to the
           current argument, and increase argv */
        arg_ptr = *argv;
        argv++;
        if (*arg_ptr == '-') /* must be an option */
        {
            while (c = *++arg_ptr)      /* get option character */
                switch (c)
                {
                case 'C' :
                case 'c' :
                    f_check = TRUE;       /* request checksum */
                    break;
                case 'D' :
                case 'd' :
                    f_sum = TRUE;         /* print checksum */
                    break;
                case 'R' :
                case 'r' :
                    f_write = FALSE;      /* read-only */
                    break;
                case 'S' :
                case 's' :
                    f_silent = TRUE;      /* a little more quiet */
                    break;
                case 'V' :
                case 'v' :
                    V_printf ("zap version 1.9\n");
                    f_verbose = TRUE;     /* a little less quiet */
                    break;
                case 'W' :
                case 'w' :
                    f_write = TRUE;       /* allow write access */
                    break;
                default :
                    error (usage);
                    break;
                }
            /* this ends the option processing */
        }
        else
        {
            /* it must be a file specification */
            file_cnt++;          /* now we've seen one */
            zap (arg_ptr);
            /* this ends the file processing */
        }
        /* this ends the argument processing */
    }
    /* if there were no filespecs, give error */
    if (!file_cnt)
        error (usage);
    /* that's it */
#ifdef vaxc
    return (1);
#else
    return (0);
#endif
}

/* current type values. note - value is also size of type */
int cur_type;
char dp_type [] = " \\/ |";
/* current display mode */
int cur_printmode;
char *defffmt[] =
{
    "0%05lo", "%6ld", "x%05lx", "0%05lo"
};
char *deffmt[]  =
{
    "0%lo", "%ld", "x%lx", "0%lo"
};
/* current file */
FILE *zf;

/* get (decimal, hex or octal) value from input line */
/* a zero return value means : ok */
int decod (buf, lp)
char *buf;
long *lp;
{
    char *cp;
    int doasc;
    int dohex;
    int dooct;
    int i;
    long num;

    dooct = dohex = doasc = FALSE;
    num = 0;
    cp = buf;
    if (*cp == ';') /* select mode */
    {
        cp++;
        if (*cp == 'x' || *cp == 'X')
            dohex = TRUE;
        else
        if (*cp == 'o' || *cp == 'O')
            dooct = TRUE;
        else
        if (*cp == 'a' || *cp == 'A')
            doasc = TRUE;
        else
        if (*cp != 'd' && *cp != 'D')
            V_printf ("input error\n");
        cp++;
    }
    else
    {
        while (*cp == '0')
        {
            dooct = TRUE;
            cp++;
        }
        if (*cp == 'x' || *cp == 'X')
        {
            dohex = TRUE;
            cp++;
        }
    }
    if (dohex)
    {
        while (isxdigit (*cp))
        {
            num = num * 16 + (isdigit (*cp) ?
                *cp - '0' : (*cp | 0x20) - 'a' + 10);
            cp++;
        }
    }
    else
    if (dooct)
    {
        while (isdigit (*cp) && *cp < '8')
        {
            num = num * 8 + *cp - '0';
            cp++;
        }
    }
    else
    if (doasc)
    {
        for (i = 0; i < cur_type && *cp; i++)
        {
            if (swab)
                num += ((long)(*cp++)) << (i << 3);
            else
                num = (num << 8) + *cp++;
        }
    }
    else
    {
        while (isdigit (*cp))
        {
            num = num * 10 + *cp - '0';
            cp++;
        }
    }
    *lp = num;
    if (!*cp)
        return (0);
    return ((*cp == '^') ? -1 : 1);
}

/* retrieve byte from file */
unsigned int gv_file (addr)
long addr;
{
    long l;

    if (fseek (zf, addr, 0))
        remark ("cannot position to %ld", addr);
    (void) clearerr (zf);
    l = fgetc (zf);
    if (l == EOF)
        remark (ferror(zf) ? "cannot read at %ld" : "read beyond eof", addr);
    return (BYTEVAL(l));
}

int tbl_max = BUF_INC;
struct ntry
{
    long addr;
    char val;
    char old;
};
struct ntry *tbl;                /* value table */
struct ntry *tbl_cur;            /* last referenced entry in table */
struct ntry *tbl_free;           /* next free entry in table */
struct ntry *tbl_ptr;            /* work pointer into table */

int locate (adr)
long adr;
{
    /* lookup address in table, return tbl_cur at correct entry
    * or next higher */
    if (tbl_cur >= tbl && tbl_cur < tbl_free && tbl_cur->addr == adr)
        return (TRUE); /* just looked up */
    for (tbl_cur = tbl; tbl_cur != tbl_free; tbl_cur++)
    {
        if (tbl_cur->addr > adr)
            break;
        if (tbl_cur->addr == adr)
            return (TRUE);
    }
    return (FALSE);
}

enter (addr, val)
long addr;
int val;
{
    char old;

    /* lookup address */
    if (locate (addr))
    {
        /* store value, if different from file value */
        if (val != tbl_cur->old)
        {
            tbl_cur->val = val;
            return;
        }
        /* else delete entry from table */
        for (tbl_ptr = tbl_cur; tbl_ptr < tbl_free - 1; tbl_ptr++)
            tbl_ptr[0] = tbl_ptr[1];
        tbl_free--;
        return;
    }
    /* if not found, tbl_cur points at next higher address entry */
    /* insert new entry at aPropriate position */
    old = gv_file (addr);
    if (val == old)              /* no-op if new == old */
        return;
    /* check for space in table, otherwise extend it */
    if (tbl_free == &tbl[tbl_max])
    {
        tbl_max += BUF_INC;
        if (!(tbl = (struct ntry *)
            realloc ((char *) tbl, (unsigned) tbl_max * sizeof (*tbl))))
            error ("table overflow");
    }
    for (tbl_ptr = tbl_free - 1; tbl_ptr >= tbl_cur; tbl_ptr--)
        tbl_ptr[1] = tbl_ptr[0];
    tbl_cur->addr = addr;
    tbl_cur->val = val;
    tbl_cur->old = old;
    tbl_free++;
}

/* retrieve value from table */
int get_value (addr)
long addr;
{
    int val;

    if (locate (addr))
        val = tbl_cur->val;
    else
        val = gv_file (addr);
    return (val);
}

/* put byte into table */
/* put value into table */
put_value (addr, val)
long addr;
long val;
{
    int i;

    for (i = 0; i < cur_type; i++)
    {
        register long temp = addr + ((swab) ? i : (cur_type-i-1));
        put_byte (temp, (int)BYTEVAL(val));
        val >>= 8;
    }
}

ptv_file (addr, val)
long addr;
char val;
{
    char c;

    c = val;
    if (fseek (zf, addr, 0))
        remark ("cannot position to %ld", addr);
    (void) clearerr (zf);
    (void) fputc (c, zf);
    if (ferror(zf) || feof(zf))
        remark ("cannot write at %ld", addr);
}

char buf [132];
char *pr_val();
long prevs [PREV_MAX];        /* previous goto table           */
int prevcnt;                 /* next free index in previous table */
```

```
push_loc (loc)
long loc;
{
    int i;

    if (prevcnt == PREV_MAX)
    {
        for (i = 0; i < prevcnt; i++)
            prevs[i] = prevs[i+1];
        prevcnt—;
    }
    prevs[prevcnt++] = loc;
}

long pop_loc ()
{
    if (prevcnt > 0)
        return (prevs[-prevcnt]);
    return (0);
}

long last_value;      /* last printed value           */
long sstart;          /* search starting value        */
long ennd;            /* search ending value          */
long interrupted;     /* search was terminated        */
int diddots;          /* dots were displayed          */

int quit_search ()
{
    interrupted = sstart;
    sstart = ennd;
}

foundit (addr)
long addr;
{
    if (diddots)
        V_printf (“\n”);
    V_printf (“Found at ”);
    V_printf (defffmt[cur_printmode], addr);
    V_printf (“\n”);
    diddots = FALSE;
    push_loc (addr);
}
```

EOF

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

SOLUTIONS TO TEST TWELVE

1. There are so many possible variations in these problems that I won't attempt to draw the circuits. Instead I'll give some of the more obvious alternatives in symmetric notation, omitting any forms which do not lend themselves readily to a quickly-drawn circuit. All networks may be drawn directly, using only basic principles, in addition to the possibilities listed below. That's not to say that you won't have discovered some neat solutions of your own among the great variety possible!

(a) S^{5}_{0},2,5 ABCDE May be drawn for level-0 and level-2, then slanted up to level-3 and shifted-down to level-1 (for 4 relays), which will give an output at level-2 for five relays.
S^{5}_{0},3,5 A'B'C'D'E' May be drawn for level-0 and level-3, and shifted-down to level-2, which will give an output at level-3 for 5 relays.

(b) S^{8}_{3},4,5,6 ABCDEFGH If drawn directly, remove a wedge of redundant contacts spanning outputs 3 - 6.
S^{8}_{2},3,4,5 A'B'C'D'E'F'G'H' Draw directly and eliminate a wedge.

(c) S^{6}_{1},4 ABCDEF Draw to level-1, and slant up to level-2 to enable a shift-down to be made to level-0.
S^{6}_{2},5 A'B'C'D'E'F' Draw to level-2 and shift-down to level-0.

(d) S^{7}_{1},4 ABCDEFG Draw directly.
S^{7}_{3},6 A'B'C'D'E'F'G' Draw to level-3 and shift-down to level-1.

(e) S^{12}_{3},10 ABCDEFGHIJKL Draw to level-3, then slant up to level-6 to enable a shift-down to level-0.
S^{12}_{2},9 A'B'C'D'E'F'G'H'I'J'K'L' Draw to level-2, then slant up to level-6 to enable a shift-down to level-0.

(f) S^{6}_{1},2,4,5 ABCDEF Draw directly and eliminate a wedge between levels 1 and 2, and between 4 and 5. OR draw for levels 1

and 2 (eliminating a wedge), shift-down to level-0 for
3 relays, and then, of course, back up to levels 1 and
2 for the fourth and fifth relays.

$S^{6}_{1,2,4,5}$ A'B'C'D'E'F' Note that the subscripts are the same as for the previous version, so the network will be exactly the same, except that the meaning of the contacts will be reversed.

Mile 16 - heading for Mile 17.

SYMMETRIC FUNCTIONS (continued)

All in all, not too bad an assignment, don't you agree? We've certainly had tougher ones to tackle along our journey! So let's get on with the

DETECTION AND IDENTIFICATION OF SYMMETRIC FUNCTIONS

Up to now we've been concentrating on m-out-of-n symmetric functions in which all the variables of symmetry have been of one type. That is, they've been either all complemented or all uncomplemented. Moreover, the specifications have been handed to us very neatly by calling for a particular m-out-of-n function. Things don't always work out so neatly in actual practice, however, and more often than not we'll arrive at some expression, such as the following. Usually this occurs after decoding a set of minterms, and we're left with the problem of deciding whether to draw a network directly from the Boolean expression, or perhaps to figure out if the expression is symmetric or not. Let's look at the expression I have in mind right now. Here it is

ab'c + a'bc + a'b'c'

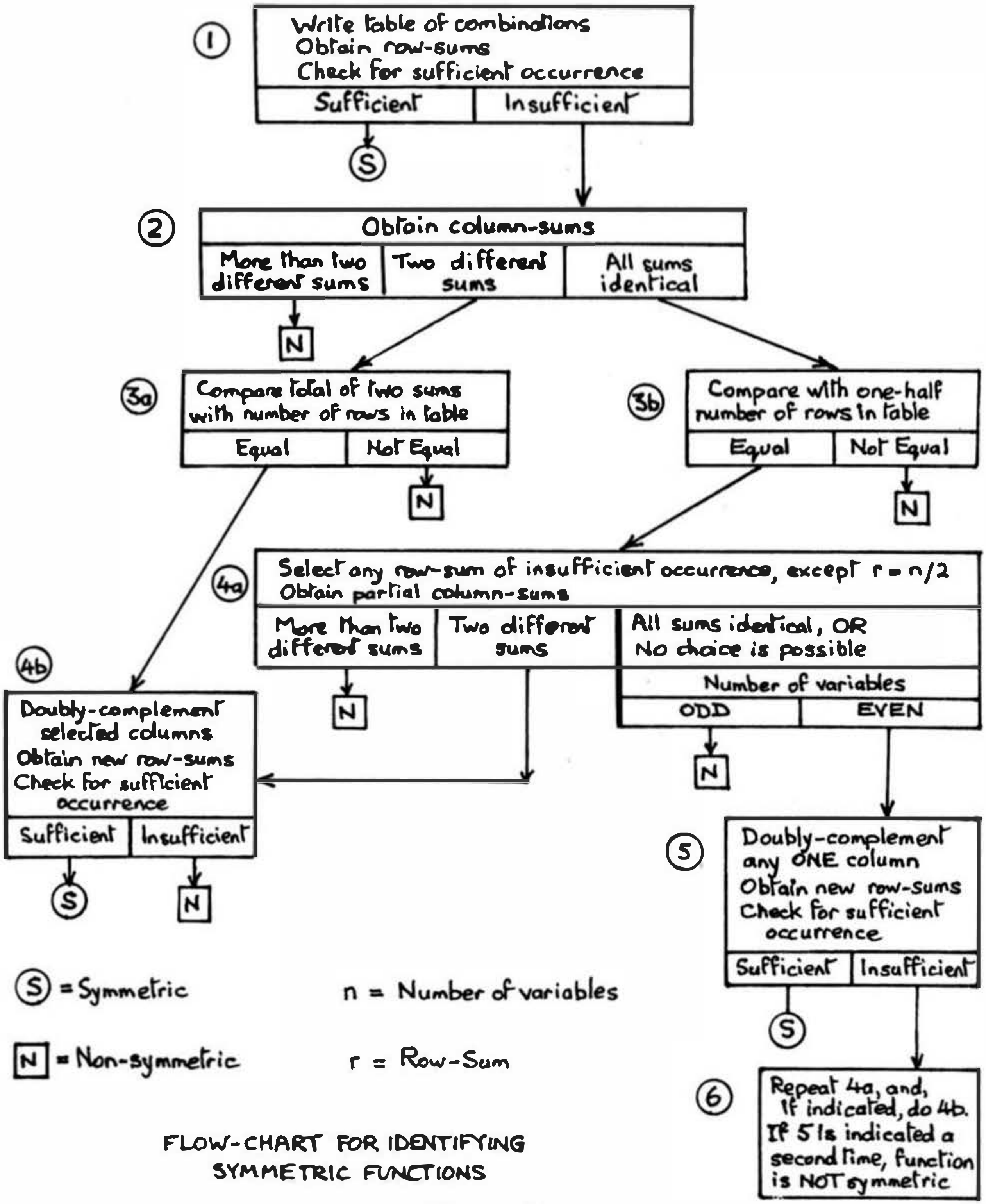
Is this symmetric, or is it not? The last term calls for an output if NO relays are operated, but unfortunately, not by any stretch of the imagination can the other two terms be said to be calling for ANY 2-out-of-3. "ac" - yes, and "bc" - yes, but NOT for "ab", so our inclination would be to say "Not symmetric".

In actual fact the function IS symmetric, in A, B and C', since interchanging ANY two of THESE variables of symmetry would leave the function unchanged. We must be careful when interchanging, say, a and c' to replace a with c', c' with a, and also the complements a' with c, and c with a'. Let's do this and get

c'b'a' + cba' + cb'a   re-arranging alphabetically to a'b'c' + a'bc + ab'c

which is EXACTLY the same as the original. Similarly with any other pair-swap! As a matter of interest, the function is S^{3}_{1} ABC' (note that C is complemented), which is a somewhat surprising result!!

What does S ABC' convey to us? The interpretation is precisely the same as before, namely that if EXACTLY one of the three conditions set out for the variables of symmetry occurs, and the other two do not occur, there'll be an output. Let's check this against the expression! Suppose only the A-condition occurred, and B and C' did NOT occur (that is B' and C did), we find that we're covered by the


1
Write table of combinations
Obtain row-sums
Check for sufficient occurrence
Sufficient
Insufficient
S
2
Obtain column-sums
More than two different sums
Two different sums
All sums identical
N
3a
Compare total of two sums with number of rows in table
Equal
Not Equal
N
3b
Compare with one-half number of rows in table
Equal
Not Equal
N
4a
Select any row-sum of insufficient occurrence, except r = n/2
Obtain partial column-sums
More than two different sums
Two different sums
All sums identical, OR
No choice is possible
Number of variables
ODD
EVEN
N
N
4b
Doubly-complement selected columns
Obtain new row-sums
Check for sufficient occurrence
Sufficient
Insufficient
S
N
5
Doubly-complement any ONE column
Obtain new row-sums
Check for sufficient occurrence
Sufficient
Insufficient
S
6
Repeat 4a, and, if indicated, do 4b.
If 5 is indicated a second time, function is NOT symmetric
S = Symmetric
N = Non-symmetric
n = Number of variables
r = Row-Sum

FLOW-CHART FOR IDENTIFYING
SYMMETRIC FUNCTIONS

Diagram 83

term ab'c. Similarly, if only the B-condition were TRUE and therefore the A and C' were FALSE (ie, A' and C are TRUE), we're covered by a'bc. Finally, if only the C'-condition were TRUE and therefore A and B were false (ie, A' and B' are TRUE), we're covered by a'b'c'.

To clinch the matter, let's interpret S^{3}_{1} ABC in the same manner (no complements this time). We have no trouble at all in deciding that if A is TRUE and both B and C FALSE, we get the term ab'c' (ie, the remaining variables of symmetry become complemented), or if B is TRUE and both A and C FALSE we get the term a'bc', and finally, if C is TRUE and both A and B FALSE we get a'b'c. The whole expression is therefore ab'c' + a'bc' + a'b'c for a 1-out-of-3 symmetric function in A, B and C. Got it now? I'd recommend you re-read the last two paragraphs VERY carefully to be reasonably certain you've got the general idea!

You'll appreciate by this time that not only is it extremely difficult to decide whether an expression is symmetric, but even assuming we've found this to be true by swapping ANY two variables of symmetry (in all possible combinations, of course) there still remains the final problem of what the subscript, or maybe subscripts, should be in the symmetric notation. Without this information we cannot even begin to design our network!

Obviously a systematic method of approach is needed, one which will not only tell us that a particular expression IS symmetric, but will also identify for us the actual symmetric function. Fortunately for us, such a method does exist! It's a very strange method, not at all difficult if you follow each step methodically, but it's well worth taking time to ponder how anyone could have developed such a system in the first place. I only wish I could claim to be that person, but alas! the facts are otherwise, and I would once more refer you to Mitchell P. Marcus' book "Switching Circuits for Engineers".

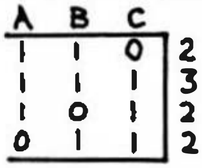
Anyway, to get back on course again, the procedure to be followed is shown as a flow-chart in Diagram 83, with a summary of the instructions to be carried out in each block of the chart. A detailed explanation of each instruction is given a little further on, with mini-examples for clarification. After this detailed explanation of all steps, a full example will be worked out to demonstrate the system, and then the inevitable TEST to exercise your own grasp of it ali.

You start in BLOCK-1, and carry on step-by-step until you end up either with an "S", indicating symmetry, or an "N", indicating non-symmetry.

DETAILED EXPLANATION OF FLOW-CHART

WRITE TABLE OF COMBINATIONS

The expression being tested for symmetry is written as a table of combinations, with each variable appearing in its UNcomplemented form at the head of a column. The simplest method is to begin with the actual minterm-numbers, INCLUDING any phis you used in your decoding (or K-map reading), as you've obviously decided to read them as 1s. These minterm numbers are translated into binary, as the example of Diagram 84 shows for minterms 6, 7, 5 and 3 (in that order) under the headings A, B and C.



A	B	C	
1	1	0	2
1	1	1	3
1	0	1	2
0	1	1	2

Diagram 84

OBTAIN THE ROW SUMS

This involves nothing more than counting the number of 1s occurring in each row, and writing the appropriate figure to the right of its row. This, too, is shown in Diagram 84.

CHECK FOR SUFFICIENT OCCURRENCE

Row-Sum	Number of Variables									
	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10
2		1	3	6	10	15	21	28	36	45
3			1	4	10	20	35	56	84	126
4				1	5	15	35	70	126	210
5					1	6	21	56	126	252
6						1	7	28	84	210
7							1	8	36	120
8								1	9	45
9									1	10
10										1

TABLE OF SUFFICIENT OCCURRENCE.

DIAGRAM 85

All row-sums are checked against the table of Diagram 85 for "sufficient occurrence". Not as frightening as it sounds! If ALL row-sums occur the required number of times, the function is symmetric and the row-sums represent the subscripts, the variables of symmetry being as shown at the head of the columns.

In the example of Diagram 84, row-sum 2 occurs three times and row-sum 3 only once. In column 3 of Diagram 85 (3 variables) we find that this is exactly the required occurrence for these row-sums, so the function is symmetric for 2,3-out-of-3, being written as $S^{3}_{2,3}$ ABC.

If any row-sum did not occur the required number of times, it wouldn't necessarily mean that the function is NOT symmetric, only that we've a little more work ahead of us before its final identification. As the flow-chart shows, we'd now head into BLOCK-2.

A DISCUSSION OF DIAGRAM 85 BEFORE PROCEEDING

The table of Diagram 85 is based on what's known as Pascal's Triangle. You can very easily extend the range to cover as many variables as you wish, by just noting that anywhere in the table that two numbers occur one above the other (except Column-10) their sum appears immediately to the right of the lower number. This should enable you to add the column for 11 variables in no time at all! Just put a "1" in row 0, followed by 11 immediately below (the sum of 1 and 10 in Column-10), then 55, and so on right down to row 10, which will have 11, and finally row 11, with another "1". Similarly for 12, 13 ....

OBTAIN THE COLUMN SUMS

Count the number of 1s in each column, and record the figure immediately below. This has been done in Diagram 86 because at least one row-sum (actually both of them) has insufficient occurrence. If more than two DIFFERENT sums occur the function is NOT symmetric. Note that we're not interested at this stage in how many times each column-sum occurs (eg "2" occurs twice), but in how many DIFFERENT column-sums there are. As we have THREE different sums (1, 2 and 3), the function depicted in Diagram 86 is NOT symmetric.

If there were exactly two different column-sums we'd proceed to BLOCK-3a, or to BLOCK-3b if they were all identical.

A	B	C	D	
0	0	1	1	2
0	1	1	1	3
1	1	1	0	3
1	2	3	2	

Diagram 86

COMPARE TOTAL OF TWO SUMS WITH NUMBER OF ROWS IN TABLE

This is an instruction to add the two different column-sums together. For example, if we had six columns, four of which had the column-sum 3, and two the column-sum 5, there are ONLY TWO DIFFERENT sums, namely 3 and 5, which, when added together, would equal 8, which is then compared with the number of rows in the table. Obviously, there are only two possible outcomes - either they're equal or they're NOT equal. In the first case, we'd be directed to BLOCK-4b, and in the other we'd know that the function is NOT symmetric.

Let's dispose of BLOCK-3b, however, before moving down to the fourth level of the flow-chart.

COMPARE WITH ONE-HALF THE NUMBER OF ROWS IN THE TABLE

Where ALL the column-sums are identical, we compare this figure with one-half of the number of rows in the table. If these two figures are not equal, the function is NOT symmetric. If they're UNequal, we get directed to BLOCK-4a, which we'll examine next.

SELECT ANY ROW-SUM OF INSUFFICIENT OCCURRENCE, EXCEPT r = n / 2

We've got this far, don't forget, because at least one row-sum didn't have sufficient occurrence. Now we can choose any such insufficient row-sum, UNLESS IT HAPPENS TO EQUAL n/2. That is, any insufficient row-sum NOT equal to one-half of the number of VARIABLES in the table. Then we make a separate little table consisting of only those rows which have our chosen row-sum. Naturally, it makes our work a little easier if we choose an "insufficient" sum contained in the least number of rows! This new table is called a "partial-table".

OBTAIN THE PARTIAL COLUMN-SUMS

Record the number of 1s occurring in each column of the partial-table. As we see from BLOCK-4a, if more than two DIFFERENT sums are obtained the function is NOT symmetric. If exactly two DIFFERENT sums are obtained, we proceed to BLOCK-4b.

Now all that remains is that all sums are identical, or that no choice was possible (see previous section) because the only row-sum of insufficient occurrence also happened to be equal to one-half the number of variables. In these two cases, if the number of variables is ODD, the function is NOT symmetric, otherwise we proceed to BLOCK-5.
But first let's dispose of BLOCK-4b.

DOUBLY-COMPLEMENT SELECTED COLUMNS

We're in this Box because we have two DIFFERENT column-sums, arriving either from BLOCK-3a or 4a, and we now have to select ONE of these sums. Just as in an earlier description, our work will be easier if we choose the column-sum which occurs the least number of times! Then we'll doubly-complement IN OUR ORIGINAL TABLE all variables (note, ALL variables) having our chosen sum. To doubly-complement means to negate (complement) the variable at the head of the column, and then complement all 1s and 0s in the column beneath. This, of course, does not change the meaning of the column, as A =1 is exactly the same as saying A' = 0. As a point of interest, if the columns in our amended table are totalled anew, ALL column-sums should now be identical.

BLOCKS 5 AND 6

The only point that needs emphasising here is

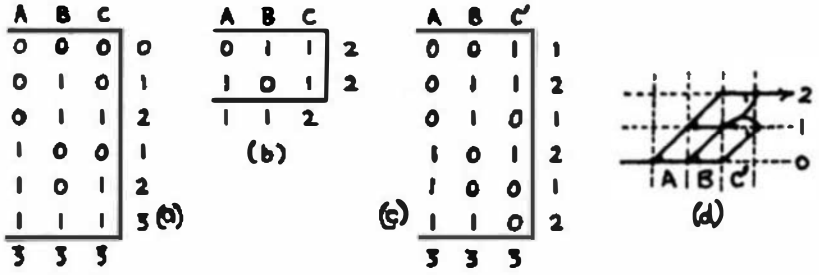
DOUBLY COMPLEMENT ANY ONE COLUMN

This means PRECISELY what it says. We are free to select any SINGLE column in our original table and doubly-complement it.

After that, the instructions are, I think, quite straightforward, and should present no difficulty in interpretation.

A FULLY WORKED OUT EXAMPLE

In the table of Diagram 87a, it's apparent that we're starting with the minterm-numbers 0, 2, 3, 4, 5 and 7, which we've written in binary under the UNcomplemented heading A, B and C. Then we obtained the row-sums.



A	B	C	
0	0	0	0
0	1	0	1
0	1	1	2
1	0	0	1
1	0	1	2
1	1	1	3
3	3	3	

(a)

A	B	C	
0	1	1	2
1	0	1	2
1	1	2	

(b)

A	B	C'	
0	0	1	1
0	1	1	2
0	1	0	1
1	0	1	2
1	0	0	1
1	1	0	2
3	3	3	

(c)

Diagram 87

Checking these row-sums for three variables against the table of Diagram 85, we find that the sums 0 and 3 have sufficient occurrence, while both row-sums 1 and 2 are insufficient. They SHOULD occur three times, but only occur twice.

We therefore proceed to BLOCK-2, where we're instructed to obtain the column-sums. They're all identical, namely 3, so we move on to BLOCK-3b, which now instructs us to compare this column-sum 3 with half the number of ROWS in our table. We have 6 rows, half of which is 3, agreeing with our column-sum. This means we must move on to BLOCK-4a.

As n/2 ("n" being the number of VARIABLES) equals 1.5, we're free to select either 1 or 2 as our insufficient row-sum. Be careful not to choose 0 or 3, as these HAVE sufficient occurrence. Let's choose 2, for no special reason, and make a partial-table of those rows with row-sum 2, as shown in 87b. Now we must count and record the partial column-sums, and as there are two DIFFERENT sums, we proceed to BLOCK-4b.

Here we're advised, though it's not mandatory, to choose the column-sum of lesser occurrence (in our case it'll be the "2" in column-C), and to doubly-complement this column in our ORIGINAL TABLE. If, say, there'd been two column-sums of 2 and three of 1, we'd have had to doubly-complement the TWO columns corresponding to the lesser-occurring column-sum 2. Back to the business in hand, we change the header-C into C' and invert all 1s and 0s in this column, as in 87c. Out of interest, we verify that all column-sums in our amended table are indeed identical. This is a good check to ensure that we haven't made a mistake somewhere along the way!

New row-sums are now obtained and checked against Diagram 85, where we find that our row-sums 1 and 2 do in fact have sufficient occurrence. This tells us that our function IS symmetric, the row-sums corresponding to our subscripts.

So we can write our symmetric function as S^{3}_{2},2 ABC' (don't forget that variable-of-symmetry-C is complemented!), and in a few more seconds we've whomped out a compact little network, and the work is done!

The network is shown in 87d, with a wedge removed in column-C'. When we come to draw a proper diagram for this network, with actual relay-contacts shown, let's NOT overlook the fact that the interpretation of diagonals and horizontals in column-C' has to be reversed from that of our more normal uncomplemented columns!

Naturally, it wasn't compulsory for me to choose variable-C for double- complementation. I COULD have chosen A and B (column-sum = 1), and doubly- complemented TWO columns instead, arriving at the symmetric function S^{3}_{1},2 A'B'C. Try it and see!! Our earlier training informs us that this is an equivalent function. Just looks different, that's all!

Whew! Took me lots longer to describe than it actually takes to carry out. So now it's your tum at the wheel! How about trying

TEST THIRTEEN

Detect and identify symmetry in the following functions. You may draw the resultant networks if you wish. "F" stands for "function".

1. Four variables.
   (a) F = 1, 2, 7, 8, 13, 14
   (b) F = 0, 5, 6, 9, 10, 15

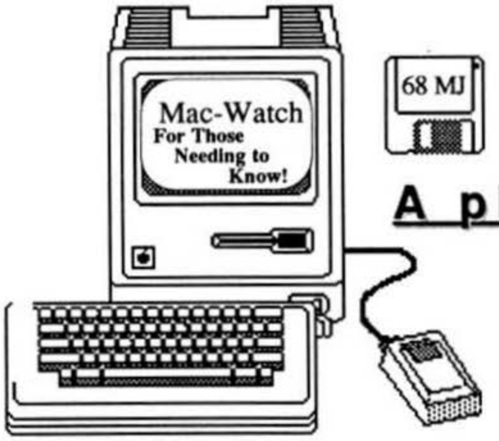
2. Five variables. F = 2, 4, 7, 14, 16, 19, 21, 26, 28, 31

3. Six variables. F = 6, 9, 18, 20, 23, 30, 33, 40, 43, 45, 54, 57

And that's really the end of symmetric functions, which are a sub-class of a larger set of functions, with even more astounding possibilities for developing the most outlandish networks you've seen in many a long year! Would you like to meet the "daddy" of symmetric functions? OK! OK! Just be patient! Next time around I'll introduce you to ITERATIVE FUNCTIONS.

... End of Mile 16. Camping at marker "Mile 17".

+++



The Macintosh™ Section

Reserved as

A place for your thoughts

And ours.......

Mac-Watch

By James E. Law
1806 Rock Bluff Rd.
Hixson, TN 37343

A Review of

Understanding PageMaker

A New Software Package for Learning PageMaker From Techware, Inc

There are a lot of companies out there (including my own) which are totally committed to 'Big Blue' and are strongly anti-Apple. Over the last 12 to 18 months, however, the tides have begun to turn and the Macintosh is being accepted as a viable business tool. One of the primary reasons for this change is the realization that the lower training costs associated with Macintosh use results in lower overall ADP costs. To a great extent, Macintosh applications can be learned by the average employee on his own.

As Macintosh applications get more complex, however, there is increased need for formal training. Even if an employee could eventually "figure out" *PageMaker*, *Excel*, or *Adobe Illustrator*, his or her company may not be able to wait that long. A number of software development companies have seen this need and are marketing software-based training on Macintosh applications. This training allows users to learn an application at their own Macintosh and at their own speed. One such offering is *Understanding PageMaker* by Techware, Inc.

Understanding PageMaker is distributed on four 800K disks labeled Novice, Novice (continued), Advanced, and Reference. A desk accessory called *HyperFormance* is included to access the various different training modules.

Understanding PageMaker is a substitute for the *PageMaker* manual. Don't expect any *Hypercard*-style interaction, sound, or animation. What you get is a 4" by 5" non-expandable window in which instructions are presented. For the most part, this window is easy to read, although some smaller size fonts are used. Frequent use is made of well designed graphics.

The size of this window bothered me at first, but later I saw it as an advantage. It contains just the amount of information needed to convey a single thought. Presentation of any more information at the same time would be confusing.

The Novice Section includes exercises which are implemented by toggling back and forth between the *PageMaker* window and the*Under-standng PageMaker* window. A series of *Understanding PageMaker* windows explains a function then gives you an exercise to try out what you have learned. Sample text and graphics are provided for placing in a *PageMaker* document. The exercises are well structured and do a good job of leading the user through creation of his first documents.

The Reference disk contains an on-screen "help" file. It enables you to quickly get information about any *PageMaker* function. Oddly enough, the windows in this section of the material do not have a button which allows you to go back to the last window. If you inadvertantly bypass a window that you want to see, you have to start over and progress to the desired window from the first window in the series.

Understanding PageMaker describes itself as "hypermedia" and allows a measure of non-sequential exploration. You route your way forwards and backwards and branch off to topics of interest through clicking a variety of buttons. It is not always clear which icons are merely illustrations and which are active buttons. The hypermedia approach helps to focus the training on your specific needs. For example, you click buttons to describe the configuration of your system (e.g., hard drive or floppy drive) and some instructions are customized accordingly. Similarly, specific instructions are customized based on whether you have *PageMaker* 1.2 or 2.0.

I progressed rapidly through several of the disks in a little over an hour each. The novice should plan to spend 2 to 3 hours per disk, however, to gain as much as possible from the instruction and exercises.

I use *PageMaker* often and consider myself to be relatively knowedgeable of its features. I learned this application by working with it until I ran into a problem then referring to the manual for a solution. The problem with this approach is that you may never learn about features that are not obvious from the menus.

This was made clear to me as I went through these lessons. Did you know that pressing the OPTION key while selecting FIT IN WINDOW causes the entire pasteboard to be displayed? Did you know that when you SAVE a PageMaker document that the old version as well as the revision to your document is saved? (The way around this is to SAVE AS rather than SAVE the final version. I tried this on two *PageMaker* documents and the size of each of them was reduced by 41%!) I did not know these items till I reviewed this software. This points out to me the importance of a structured approach to learning software, even for those who think that they have some knowledge of the program.

This package can serve as a useful tool for learning *PageMaker*. You will still need an "expert" to call on when you get stuck but, for the most part, *Understanding PageMaker* may be the only teacher you need.

Techware, Inc. must be out of touch with the main stream of Macintosh software development in that *Understanding PageMaker* is copy protected. You are able to copy it onto your hard drive but the original disk will be demanded each time you start up. Many users will find this to be a major hindrance to their use of this software.

In conclusion, *Understanding PageMaker* is a simple but effective substitute for the *PageMaker* manual. It provides a thorough, clear, and structured approach to learning this software. The overall value of this package is seriously diminished, however, by its antiquated copy protection scheme.

EOF

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01453

LIVE AND LEARN

That appears to be the story of my life with FORTH, and I love it! I don't know how many of you have noticed it, but my programming habits have changed a lot in the last couple of years that I have been writing this column. For one thing, it has forced me to think about why I do certain things the way I do them and to look for better ways.

It has been said that the best way to learn something is to try to teach it to someone else. I think that this column has had that effect on me; gradually but surely, I have been learning more about FORTH and how it works. I still have a lot to learn, but I now think that I can be safely considered as an intermediate level FORTH programmer.

For that, I would like to thank all of you; as well as the patient people at 68' MicroJournal for all of you help. Particularly, I would like to thank Wilson Federici for his great software!

No, this is not a goodbye, I just wanted to say "thank you" to the people who have been so nice to me!

KEYBOARD NUMBER INPUT

Back in the January, 1987, issue, I wrote a rather elaborate set of definitions which would fetch a number from the keyboard. There was built-in error trapping, which protected the user pretty thorougly from entering an unacceptable key as a digit. Well, I found that I really never used these definitions very much, except in finished programs, because I normally did not need that much error trapping while I was developing an application. I really wanted something much simpler, so I would usually scratch up some definition on the spot.

However, I have now settled on the definition I call @# ("fetch number"). It is really a simplification of those earlier definitions, and it has no elaborage error trapping. The only error trapping is the program crash you get from NUMBER if you enter an impossible key. Simplicity, not speed, was the driving force behind this definition.

As you can see from the first line, @# will accept only one keypress before processing the entry. This was deliberate in order to eliminate the need for pressing the <enter> key.

The remainder of the first line puts a space after the input "string" to set it up for NUMBER . The first part of the second line does the conversion or crashes the program, depending on the results of the conversion.

The final DROP is simply to remove the 0 from the Data Stack which was put there by NUMBER when the conversion was originally made to a 32-bit number.

There it is; simple, but not very elegant!

FORWARD ADDRESS REFERENCES

The subject of forward referencing in FORTH came up at one of our local FIG chapter meetings in a rather interesting way. A member asked if there was an easy way to call menus from other menus; none of his efforts would compile.

A little reflection makes it obvious that this should be a classic case of needing a forward address reference; but how can you do that in a language with only a single-pass compiler? Herein is one answer.

Though I know that this is not the only way to solve this problem (as is nearly always true in FORTH), I think that my way has the following advantages:

(1) portability, (2) easy to write, and (3) easy to understand. As I think about it, (3) may be the most important advantage!

Since you cannot make a forward address reference in FORTH, you must trick the compiler into thinking that the address has already been defined, even though it has not. One of the best ways to do this is to use a jump-vector table. This is a fairly common practice in assembly language programming, but not encouraged much of anywhere else; probably because it is a virtual GOTO statement. The GOTO statement, in whatever form, is frowned upon in polite society these days, but it is sometimes the only way to solve a problem without a lot of crazy complications.

A jump-vector table is nothing but an array of addresses. You use it by indexing into the array as far as the address you are interested in, read that address, and jump to it. It is a form of indirect addressing, since the address the compiler is interested in is the address of the storage slot and not of the contents of the slot!

In other words, you can refer to the address of a slot within a table anytime you like (after the empty table has been defined) without ever having to know what is in that slot. Of course, you must eventually put something into that slot, or your program will jump to never-never-land if it ever tries to access that address. Because you can refer to the address of the slot without having to know its contents, the compiler can process a call to the table without any problems!

Before going any further, let's do a little housekeeping. Later on, we will need to process an input error, so let's get that out of the way right now. I hate programs that tell me that I have made an error, but don't tell me what that error is. SELECTION-ERROR tells you that an error has been made and what that error is. This is the place that you put any sort of error processing that you believe necessary. You can be as simple or as elaborate as you like.

The real meat of the application begins with the next line. Here we define and initialize the vector-table array to be named MENU-LIST . The 3 is the number of vector addresses to be accomodated by the table and the 2* multiplies this number by 2, since all addresses need 2 bytes. ALLOT reserves the space.

I must confess that DO-IT may not be the best name for this definition, but it sure describes what the definition does! DO-IT is a factor which is used 9 times in the following definitions; it actually finds the execution vector and then jumps to it.

DO-IT is entered with the vector "offset" into the table already on the Data Stack. When this number is multiplied by 2 and added to the address returned by MENU-LIST , you have a pointer to the desired execution address. This address is brought to the Data Stack and executed.

A casual inspection of the definitions for MENU0 , MENU1 , and MENU2 may not show where they are diferent; however a more careful reading of each definition will show several differences. I debated whether or not to factor out the common lines, but decided that that could be more confusing than the present situation.

The differences are in the selection table and in the CASE ... ENDCASE structure. In each definition, only the other two menus are mentioned; the current menu is ignored.

The first 6 lines of each definition print the title of the menu and the choices available. Line 7 asks for your choice and uses @# to get your answer. You would want to use something safer for a real application, but @# is adequate for this demonstration.

The input from @# is then filtered through the CASE ... ENDCASE structure for the appropriate action. If you enter a number outside the range of 0-3, SELECTION-ERROR will trap it and the phrase 0 DO-IT will throw you back to the first menu. This lets you start over in the menu path without a system crash or the need for recursion; now this is simple and elegant!

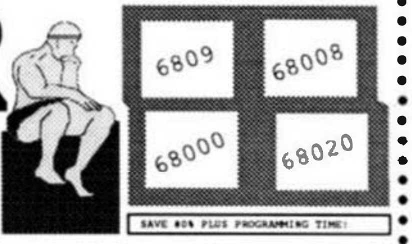
This is one of the most appreciated features of CASE ... ENDCASE . The automatic error trapping is always there waiting for you to use it, and the trapping can be as minimal or as thorough as you want to write. What more could you want?

Don't forget to load the jump-vector table now that the menu has been defined. You can wait until all of the menus have been defined and then stuff the table like the proverbial Christmas goose, or you can add each vector as soon as it is available. Suit yourself; just don't forget!

I chose to have the format of the three lines the same for loading the vector table. However, it should be obvious that there is some simplification possible for MENU0 and MENU1 , but I prefer to leave well

Continued On Page 36

SCULPTOR



From the world's oldest & largest OS-9 software house!

CUTS PROGRAMMING TIME UP TO 80%

6809/68000-68030 Save 70%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develope DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.6	6809 - $1295.00
	68000 $1295.00
	68020 $1990.00

Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!

Once this supply is gone - the price goes back up!

System OS-9: 6809/68000-68030

- **Regular ~~$1295.00~~**

+ $7.50 S&H USA
Overseas - Shipped Air Mail Collect

ONLY $295.00

S.E. MEDIA
POB 849
5900 Cassandra Smith Road
HIXSON, TN 37343 615 842-4601

$AVE - WHILE SUPPLIES LAST!

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTWARE

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE

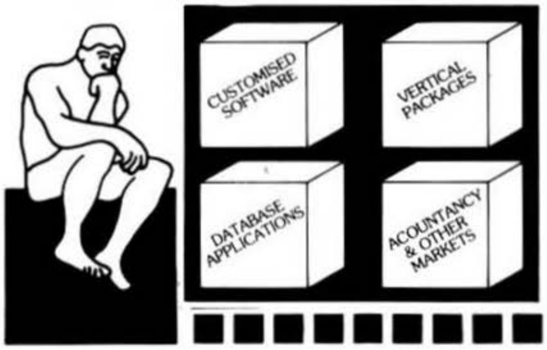
With every development system you receive

□ A manual that makes sense
□ A periodic newsletter
□ Screen form language
□ Report generator
□ Menu system
□ Query facility
□ Set of utility programs
□ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts ■■■■■ **Features** ■■■■■■■■

Sculptor for 68020
OS-9 & UniFLEX
$995



DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from

□ alphanumeric
□ integer
□ floating point
□ money
□ date

DATA FILE STRUCTURE

□ Packed, fixed-length records
□ Money stored in lower currency unit
□ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

□ automatic by field type
□ validation list in data dictionary
□ programmer coded logic

ARITHMETIC OPERATORS

−	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
−	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES

□ Full date arithmetic
□ Echo suppression for passwords
□ Terminal and printer independence
□ Parameter passing to sub-programs
□ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS

□ Define record layout
□ Create new indexed file
□ Generate standard screen-form program
□ Generate standard report program
□ Compile screen-form program
□ Compile report program
□ Screen-form program interpreter
□ Report program interpreter
□ Menu interpreter

□ Query facility
□ Reformat file
□ Check file integrity
□ Rebuild index
□ Alter language and date format
□ Setup terminal characteristics
□ Setup printer characteristics

SCREEN-FORM LANGUAGE

□ Programmer defined options and logic
□ Multiple files open in one program
□ Default or programmer processing of exception conditions
□ Powerful verbs for input, display and file access
□ Simultaneous display of multiple records
□ Facility to call sub-programs and operating system commands
□ Conditional statements
□ Subroutines
□ Independent of terminal type

MUSTANG-020 Users - Ask For Your Special Discount!

	*	**	***		*	**	***
MUSTANG-020	*$1,990	$398	$795	**PC/XT/AT/MSDOS**	$695	$139	$299
MUSTANG-08	*$1,295	$259	$495				

Call or write for prices on the following systems.

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX,REGAL), STRIDE, ALTOS, APRICORT, ARETE, ARM-STRONG, GLEASDALE, CHARLES RIVERS, GMX, CONVERG.TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N.STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6809/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

* Full Developement Package
** Run Time Only
*** C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex: 5106006630

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.

F, S, CCF - $99.95

Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.

Special -- CCF $35.00; F, S $50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK*DOS.

*FLEX, SK*DOS, CCF, OS-9 $99.00*

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.

F, S, CCF $150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.

F, S, CCF - $75.00

XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8

F, S, CCF - $98.00

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00 - CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00
68010 Disassembler $100.00 F, S, O, U, UNIX, MS-DOS

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95
F, S, " " $100.00 - O, object only $150.00
U, " " $300.00

CROSS ASSEMBLERS

CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.

*68000 or 6809, F, S, CCF, O, U, *Macintosh, *Atari, UNIX, MS-DOS*
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - w/source $500.00

XASM Cross Assemblers for FLEX, SK*DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.

*Complete set, FLEX, SK*DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
SK*DOS	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.

*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

*FLEX, SK*DOS, CCF, OS-9, UniFLEX, UNIX, MS-DOS, 68000 & 6809 with Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) $99.95
X-TALK Software (2 disks only) $69.95
X-TALK with CMODEM Source $149.95

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.

U - $299.99

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

F, S, CCF - $198.00

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.

*FLEX, SK*DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers.* "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

F, S and CCF - $195.00

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK*DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

F, S and CCF - $295.00

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.

*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

F, S and CCF 5" - $190.00 F, S 8"- $205.00

OmegaSoft PASCAL from Certified Software -- Extended Pascal for systems and real-time programming.

Native 68000/68020 Compiler, $575 for base package, options available. For OS/-9/68000 and PDOS host system.

6809 Cross Compiler (OS-9/68000 host) $700 for complete package.

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*

*FLEX, SK*DOS, CCF - $99.95*

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - $58.95

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.

FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.

Like compilers for other languages, FORTHBUILDER can operate in "batch mode".

The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.

FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.

F, CCF, S - $99.95

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

* Now supplied as a two disk set:

Disk #1: JUST2.CMD object file.

*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*

Disk #2: JUSTSC object and source in C:

*FLEX, SK*DOS - OS9 - CC*

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p ,u ,y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only- F, S & CCF - $49.95
Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

*Regular FLEX, SK*DOS $129.50*
** SPECIAL INTRODUCTION OFFER * $79.95*
SPECIAL PAT/JUST COMBO (w/source)
*FLEX, SK*DOS $99.95*
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- **OVER 150,000 words!** *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,
F, S or O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,
F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - $59.95,
F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

F, S or O - $329.95, U - $549.95
O, 68000 $695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems
FOR 6809 FLEX-SK*DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 3% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

ITS EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

FOR 6809 FLEX-SK*DOS(5/8") **$249.95**

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

O & CCO obj. only - $89.95

Luckdata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

F, S, CCF --- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media -- **A UniFLEX BASIC decompiler** Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.

U - $219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK*DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09.
6 were written in C Language and the remainder in assembly.
All the routines are compiled down to native machine code which makes them fast and compact.

1. **CFILL** -- fills a string with characters
2. **DPEEK** -- Double peek
3. **DPOKE** -- Double poke
4. **FPOS** -- Current file position
5. **FSIZE** -- File size
6. **FTRIM** -- removes leading spaces from a string
7. **GETPR** -- returns the current process ID
8. **GETOPT** -- gets 32 byte option section
9. **GETUSR** -- gets the user ID
10. **GTIME** -- gets the time
11. **INSERT** -- insert a string into another
12. **LOWER** -- converts a string into lowercase
13. **READY** -- Checks for available input
14. **SETPRIOR** -- changes a process priority
15. **SETUSR** -- changes the user ID
16. **SETOPT** -- set 32 byte option packet
17. **STIME** -- sets the time
18. **SPACE** -- adds spaces to a string
19. **SWAP** -- swaps any two variables
20. **SYSCALL** -- system call
21. **UPPER** -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
Limited Special - $19.95

SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK*DOS files.

COMMON compare two text files and reports differences.

COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK*DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

ECHO echos to either screen or file.

FIND an improved *find command* with "*pattern*" matching and wildcards. Very useful.

HEX dumps files in both hex and ASCII.

INCLUDE a file copy program that will accept "includes" of other disk files.

KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.

LISTDIR a directory listing program. Not super, but better than CAT.

MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.

MULTICOL width of page, number of columns may be specified. A MUST!

PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.

REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!

SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.

SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on n* word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.

TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.

TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.

UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.

WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants - - TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F, S and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/ Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk)** can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file,** except they have the extension '**.DIR**'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS , FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
F, S and CCF 5" - $50.00   F, S 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
6809 O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label defuiitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
ALL Utilities include Source (either BASIC or A.L. Source Code).
F, S and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

MS-DOS-FLEX Transfer Utilities to OS-9 For 68XXX and CoCo* OS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.
**CoCo Version: $69.95   68XXX Version $99.95*

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.
U - $395.00, F, S, OS-9 and SPECIAL CCF - $250.00
OS-9 68K - $299.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
F, S - $59.95, U - $89.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK*DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F, S and CCF - $79.95

NEW

MS-DOS/FLEX Transfer Utilities For 68XXX and CoCo* OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.
CCO $69.95   68XXX OS-9 $99.95

Macintosh Software at Discounted Prices

"Call for prices, it"ll be worth the savings."

NOTE: Changes

1. Price increase for SCULPTOR, see advertising front of this catalog and other ad in this issue. Special price for 68 Micro Journal readers.
2. Lower price for BASIC09 TOOLS, see Utilities section.
3. New MS-DOS & FLEX to OS-9 Utilities, see above.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

enough alone. These lines are only executed once, during compilation, and never again, so there is hardly any speed penalty to be concerned with. I think that having the lines all the same makes it easier to spot errors, and I always appreciate that.

Notice the ' (tick) at the first of the line. Be sure that you enter it when you type the line: otherwise the system will crash the first time you try to call that menu.

Any one should be able to take this skeleton structure and modify it to fit a specific situation. Certainly, more information on how to make a selection could be useful. Also, a real application would do other things besides simply select the other menus. Enter these definitions, try them out, use them as "boiler plate", and, most important of all, add your own variations.

Until next time, may the FORTH be with you!

```
: @#  ( - n )
\ RDL 08/19/88
\ Input a single digit
    PAD  1  EXPECT    BL  PAD
1+  C!
    PAD  1-  NUMBER    DROP ;


: SELECTION-ERROR ( - )                           \ RDL 08/19/88
\ Error message
    CR  ." ERROR-INVALID CHOICE"  CR ;


CREATE  MEND-LIST  3  2*  ALLOT                    \ RDL 08/19/88
\ Initialize vector array


: DO-IT ( n - )                                   \ RDL 08/19/88
\ Process execution vector
    ( n )  2*  MENU-LIST  +             \ point to vector
    @  EXECUTE ;                        \ make vectored jump


: MENU0 ( - )                                     \ RDL 08/19/88
    CR  CR  ." This is the FIRST menu."  CR
    ." Select one of the other menus by pressing its seclection"
    ."  code."  CR
    ."      0  EXIT from this program."  CR
    ."      2  SECOND menu"  CR
    ."      3  THIRD menu"  CR
    ." Your choice: "  @#
    CASE
    2  OF  MENU-LIST  1  DO-IT  ENDOF
    3  OF  MENU-LIST  2  DO-IT  ENDOF
    0  OF  CR  ." Graceful exit."  CR  ABORT  ENDOF
    SELECTION-ERROR    0  DO-IT
    ENDCASE ;

'  MENU0  MENU-LIST  0  2*  +  !          \ load vector table

: MENU1 ( - )                                     \ RDL 08/19/88
    CR  CR  ." This is the SECOND menu."  CR
    ." Select one of the other menus by pressing its seclection"
    ."  code."  CR
    ."      0  EXIT from this program."  CR
    ."      1  FIRST menu"  CR
    ."      3  THIRD menu"  CR
    ." Your choice: "  @#
    CASE
    1  OF  MENU-LIST  0  DO-IT  ENDOF
    3  OF  MENU-LIST  2  DO-IT  ENDOF
    0  OF  CR  ." Graceful exit."  CR  ABORT  ENDOF
    SELECTION-ERROR    0  DO-IT
    ENDCASE ;

'  MENU1  MENU-LIST  1  2*  +  !          \ load vector table

: MENU2 ( - )                                     \ RDL 08/19/88
    CR  CR  ." This is the THIRD menu."  CR
    ." Select one of the other menus by pressing its seclection"
    ."  code."  CR
    ."      0  EXIT from this program."  CR
    ."      1  FIRST menu"  CR
    ."      2  SECOND menu"  CR
    ." Your choice: "  @#
    CASE
    1  OF  MENU-LIST  0  DO-IT  ENDOF
    2  OF  MENU-LIST  1  DO-IT  ENDOF
    0  OF  CR  ." Graceful exit."  CR  ABORT  ENDOF
    SELECTION-ERROR    0  DO-IT
    ENOCASE ;

'  MENU2  MENU-LIST  2  2*  +  !          \ load vector table
```

+++

SK*DOS and the PT68K-2 A FUN Way To Learn 68000 Computing

By: Michael Daly
334 Main Street Apt. B-4
East Greenville, Pa. 18041

A computer and associated software that is intended for the hobbyist should be affordable, powerful, rational in design and FUN to use. After working for years with computerized x-ray scanners using minis which were powerful but certainly not affordable or practical for hobby use and IBM PC's which were affordable but no fun, I've found the system I've always wanted in the PT68K-2 68000 computer running SK*DOS as the operating system.

The PT68K-2 is a single board computer available as either a kit or fully assembled with several configuration options from Peripheral Technology that combines performance and affordability. SK*DOS is an operating system that has the power to take advantage of the computer's capabilities without being cumbersome or complex. Together they combine to become a highly useful, cost-effective and FUN introduction to 68000 computing.

I purchased my PT68K-2 as a kit running at 12.5 mhz with 1 meg of ram, monochrome monitor, two 720k 5.25" drives and the usual items (power supply, case, enhanced keyboard,etc.) to make a complete system. Building the unit was done in stages so that construction errors could be more readily traced and corrected. The cpu support circuits went in first, then address, ram and i/o circuits. Usually it would not be practical to build a complex device as a computer under home conditions but due to the modular implementation and use of the supplied HUMBUG monitor the computer was up and running in three days (working on it only a couple of hours per day) using nothing more complicated than a multimeter and a logic probe. The kit comes with assembly instructions that are sufficient for the advanced kit builder and the PT68K-2 theory of operation has been the subject of a multi-part tutorial by Peter Stark.

After the first few days of use, however, a curious thing happened. The system would go into never-never land and could only be brought back by a hardware reset. This would occur more frequently with some software and not at all with other programs. I should mention at this point that the system board has a mpu clock jumper so the user may select either a 8mhz rate or the optional faster rate (10 or 12.5 mhz.). When the 68000 cpu clock was set to 8mhz the system was rock steady with no problems of any sort. I had become so hooked using the system (a MOST re freshing change from the Intel-based systems I'd used in the past) that I couldn't bear to part with it long enough to send it back to Peripheral Technology for their evaluation. Finally, while waiting for an update of my SK*DOS disk to arrive, I sent the board back to PT and after much head scratching on their part, they got the board to function reliably at 12.5mhz. This was accomplished by changing the 74LS10 in the dtack circuit to a 74S10 and changing the value of one of the capacitors. The unit works great but Fred Brown of PT can't explain why my board was so fussy when all the other units shipped were OK (since he designed the board, I'm not even going to try to explain it). I just want to say many thanks for his efforts - it's nice to know you've given some of your hard earned cash to a company that stands behind their product and in my case went the extra distance to provide customer satisfaction. Along with the speed fix I also acquired a 40meg hard drive, 1200 baud modem and a printer to round out the hardware part of the system. Since the PT68K-2 uses IBM clone parts, these additions were accomplished at a very reasonable cost.

Computer hardware is useless, of course, without control and application software to run it. The operating system can either enhance the hardware or degrade it's performance by being so cumbersome or complicated that the average (read non-professional programmer - like me) can't access all the system features. Being awkward or complex to the point of obscurity is NOT the case with SK*DOS.

This is a single-user, single-tasking OS (for now) that comes with a trove of utilities that are simple to use and very functional (not just bells & whistles). SK*DOS is produced by Peter Stark's Star-K Software Systems and comes bundled with the PT68K-2 when the floppy drive kit option or PT assembled board is purchased.

It supports custom device drivers, batch files, keyboard typeahead, i/o redirection, pipes, TSR's, memory caching and a ramdisk. The utilities have built-in help messages and are straightforward in their use. Want to format a floppy or winchester drive? No problem, just call the appropriate format program, answer questions regarding the drive's specifications and there you have it - a formatted drive with the bad blocks deleted from the free chain (if it's a winchester partitioning is also accomplished at the same time in accordance with your responses to the formatter's queries). Try doing the same thing with a ms-dos based system! It's a JOKE!! All the other utilities are as easy to use. Those who have been following Ron Anderson's column know that SK*DOS provides substantial help for those who work in assembler. Sixty DOS calls and seventeen ROM calls are available along with a file of system equates to make assembly easier.

SK*DOS is highly similar to FLEX and those familiar with that OS should feel right at home. In fact, I've heard SK*DOS described as the OS that continues where FLEX stopped. Peter is responsive to feedback from users and even if he doesn't agree with your view at least he'll listen. If he likes an idea, it'll be incorporated in a revised version. His update policy is as rational as the OS - send back your disk along with return postage and your update is on it's way. Peter's common-sense design of SK*DOS together with his acknowledgement of customer feedback add up to a class act that the big boys who write more mainstream OS's would be wise to emulate.

An operating system without application programs would be as useful as computer hardware without a power supply. Fortunately, the PT68K-2 can run the vast majority of software written for FLEX as SK*DOS has as a utility a 6809 emulator. The SK*DOS users group under the guidance of Sidney Thompson can provide a number of 68000 programs such as MicroEMACS, small-C compiler, communications software and other programs all for a nominal distribution charge. Bundled with SK*DOS is an assembler written by Computer Systems Consultants. When you purchase the full K & R C compiler from CSC you also get the assembler and MicroEMACS. Other vendors are beginning to implement or have already implemented software such as spelling checkers, disassemblers, C programming support tools and MIDI control programs. All of these programs are quite reasonable in cost and shareware/freeware applications can also be acquired.

Alright, by now you're convinced I'm a relative of Fred Brown or Peter Stark (NOT SO!!). There are things about the system that I'd like improved - nothing's perfect. Part of my wish list would be to increase memory beyond 1 meg, high-res graphics, standard implementation of subdirectories, better management of hard drive backup/ restore and so forth. But the design concept and it's implementation is sound only the maturity that comes with a large user base is lacking. The foundation is there to build on and it's strong enough to carry the advanced applications of tomorrow. In the meantime, I'm going to continue having fun while I learn 68000 programming techniques and increase my knowledge of computer hardware.

The SK*DOS USER's Group can be reached c/o Sidney Thompson 181 Greenbriar Ct., Conyers,Ga. 30208 (phone 404-922-3097, eves., voice)

Star-K Systems operates a 24 hour bbs at 914-241-3307 and is a useful forum for exchanging ideas and software.

Michael Evenson runs a 6800(0) bbs at 817-488-8398. Mike has contributed a number of utilities for the PT68K-2 and lists files for OS/9 and other OS's on his board as well and is most helpful in answering questions you may have regarding 68000 computing.

Peripheral Technology, Inc.
1480 Terrell Mill Rd. Suite 870
Marietta,Ga. 30067
404-984-0742

PT68K-2 Specifications

CPU - Motorola MC68000
Clock - 8mhz standard, 10 or 12.5mhz optional
Ram - 512 - 1024k 0 wait state with 4k static/battery backup
ROM - 32 - 128k EPROM
Serial - Four RS-232 ports
Parallel - two 8 bit ports
Real-time clock/calendar with on-chip battery
Expansion - Six IBM PC-compatible i/o slots (memory cards not supported)
Floppy - up to four with on-board WD1772 controller
Hard Disk - ST-506 compatible up to 64 megabytes per drive
Console - serial terminal or mono/cga monitors with IBM type keyboard
Power - 150 watt IBM plug compatible
OS - SK*DOS standard, OS/ 9 $500.00 option
Configuration - kit or fully assembled & tested - contact Peripheral Technology for latest prices and complete details

+++

UniFLEX Internals

Egbert Jan van den Bussche
Raam 50-a
2611 LV DELFT
HOLLAND

PART 1

Introduction.

UniFLEX(tm) is a multi-user/multi-tasking UNIX(tm) like operating system for 6809 and 680XX processors written by TSC. It was originally developed for SWTPc 6809 hardware but the 680XX version is ported to many other brands. The system is completely written in assembler and is therefore very fast. I'll try to describe some changes made to 6809 UniFLEX to adapt it to different hardware. I'm afraid this story will be a little of this and a little of that, just because I don't know where to start. If I get requests from readers for special items I'll be happy to dig into it and steer the story in that direction. However, I must admit that UniFLEX has still a few black holes for me too...

How it all started.

I followed the whole live of UniFLEX from the very first version shipped. In those years I worked for a company representing SWTPc and TSC in Holland. We were using FLEX on 6809's and multi-user SDOS (Software Dynamics) on MSI systems (Midwest Scientific Instruments) at that time. We're talking pre-wincester days, end seventies. When UniFLEX first came in we were running it on dual floppy systems (SWTPc 6809/DMF-2) and were amazed how fast the drive select LEDs could flicker (amongst other more useful features...). I think it was even a 1 MHz system, yes, must be, because soon after we got the CDS-1 and we had a lot of trouble to get it to run on 2 MHz. I remember spending X-mas day and even old years evening in the office. Later we received the add-on board for the MD-HD board, and finally it worked. Then came the small winchesters (DMF-3) which were much cheaper but, at the same time the dollar became so expensive that we lost the fight against the PC, and finally we had to cease business...

About the hardware.

At that time I had quite a lot of SWTPc hardware at home, mostly 6809 FLEX systems that were traded in for 20-bit address systems (S/09). Being used to UniFLEX, I wanted it at home also and I started building a DMF-2 controller, bought 2 8" floppy drives, converted a MP-B2 motherboard to extended addressing and added the functionality of the MP-ID (timer/baudrate parallel printer port) on the I/O bus. A lot of wire wrapping later, the system behaved as a much newer SWTPc box. I went a little bit further by making the I/O decoding more precise, SWTPc had always a lot of double addressing on $XE000 and $XF000, throwing away nearly 128 K of address space. I had big plans for those address ranges like virtual disk and a ROM area. Because nobody with standard hardware would be using this address space, I was sure of an interference free private playground. Well, the RAM disk is still not there but recently I exchanged a 64K RAM board (by the way: this board is very easily converted to 256 K...) for a 64K ROM/RAM board which will be put to work soon. I'm sure it must be possible to put UniFLEX in ROM and start it from there with a UniBUG command.

Putting efforts together.

In the mean time some 6809 fanatics with similar systems gathered together and founded the CS/SWTPc user group and UniFLEX was ordered by us for some of the systems that should meet the requirements, and in fact it WORKED!. Then the fun started. As real hobbyists we wanted things different than standard. We started adding a winchester drive, not the SWTPc way with the WD100X controller but with a SASI controller which happened to be available. This enforced us to find out how UniFLEX handles its devices. We disassembled it and found ourselves confronted with a 250 page 'source' listing. A lot of days (and nights) later we managed to overlay the original CDS harddisk driver in UniFLEX with our own code and /dev/hd0 came to life. Formatting the thing was done by sending the correct bytes to the SASI controller with monitor routines. The SASI controller is slightly more intelligent than the Western Digital controller, it can format the drive quite easily. At the time of this writing we do not use overlaying any more we just relink a bunch of relocatable modules.

Modifications to my home system.

I sticked to the TSC/SWTPc approach and piggy-backed yet another board on my DMF-2 controller (See 68 Micro Journal...) as host adapter to the WD1002-5 controller board I bought. This controller supports 3 winchesters and 4 5.25 floppy drives. The host adapter is connected to the DMA controller on the DMF-2 controller similar to the situation on the DMF-3 board, but there are certain hard to avoid differences like inverted data path to floppy controller chip. The interrupt trapping is also slightly different. In figure 1 you see the host adaptor for the WD1002-5 board. It's more or less the same as on the DMF-3 board. The DMA controller still resides at address $FF000-$FF01F, the floppy controller at $FF020-$FF023, at $FF024 is the drive select latch located, the WD1002 board is addressed from $FF030-$FF036, at $FF040 the extended addressing latch and at $FF070 is an interrupt source latch added. You see the DMF-2 board is a workhorse in my system, it services 2 8" floppies, 2 5.25" floppies and, via the WD1002, one 15 Mb winchester, two 5Mb winchesters and a 80 track 5.25" floppy (not operational yet).

On the 30 pin bus are located: 6 SI-1 (MSI) serial interfaces ($FE000/04, $FE010/14,$FE020/24), a home-made MP-QP at $FE060 combined with a serial printer port at $FE070. The MP-ID substitute with timer ($FE090) and parallel printer port ($FE080) and a clock/calendar board ($FE0A0) are all three on one board on port 7. The monitor ROM on the CPU board is essentially UniBUG 1.9, but my version accepts a drive number after typing 'F' or 'W' to boot from floppy or winchester.

Software.

To get UniFLEX work with this configuration a few changes were made.

1. A 'formatwd' was created from the standard 'formatwd1000'. TSC uses for all formatters the same program, but every time a different 'write track routine', 'set parameters routine' and the correct boot code.

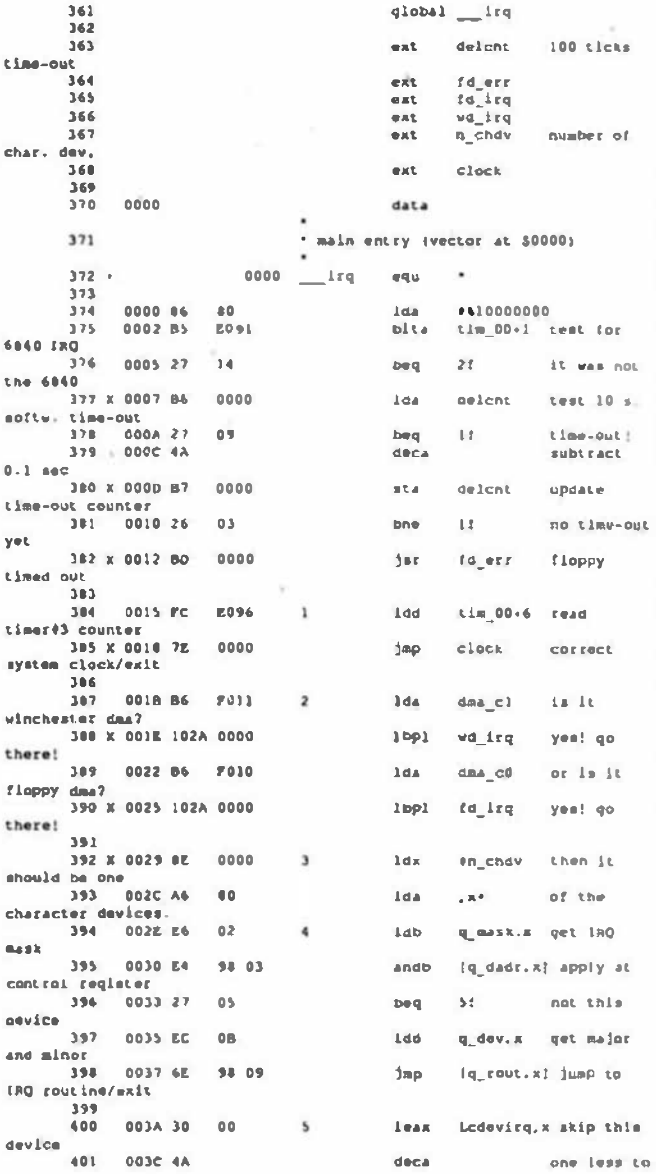
2. A new wd driver and IRQ routine for UniFLEX. UniFLEX must configure itself to use the correct boot drive.

See listing 1 for my interrupt routine. I check only for ACIA like terminal ports, floppy or winchester IRQ and no harddisks (CDS-1 like) at $FF100 or $FF300. Somewhere near the end the IRQ source is checked and a jump is made to the correct interrupt handler.

Listing 2 gives the source code for the winchester bootstrap routine. It's quite obvious what happens if you know how the file system works. I'll come back to this later on.

Listing 3 is one of the UniFLEX initialization routines where ROOT, PIPE, and SWAP are assigned to logical devices. This routine doesn't use the settings inside UniFLEX (see /etc/tune) but copies the information set up by the monitor ROM to those internal locations.

Next time some more information on the file system.



```
                                      global  __irq

                                      ext     delcnt     100 ticks
time-out
                                      ext     fd_err
                                      ext     fd_irq
                                      ext     wd_irq
                                      ext     n_chdv     number of
char. dev.
                                      ext     clock

         0000                         data
                              *
                              * main entry (vector at $0000)
                              *
                        0000  __irq   equ     *

         0000 86   80                 lda     #%10000000
         0002 B5   E091               bita    tim_00+1   test for
6840 IRQ
         0005 27   14                 beq     2f         it was not
the 6840
       x 0007 B6   0000               lda     delcnt     test 10 s
softw. time-out
         000A 27   09                 beq     1f         time-out!
         000C 4A                      deca               subtract
0.1 sec
       x 000D B7   0000               sta     delcnt     update
time-out counter
         0010 26   03                 bne     1f         no time-out
yet
       x 0012 BD   0000               jsr     fd_err     floppy
timed out

         0015 FC   E096      1        ldd     tim_00+6   read
timer#3 counter
       x 0018 7E   0000               jmp     clock      correct
system clock/exit

         001B B6   F011      2        lda     dma_c1     is it
winchester dma?
       x 001E 102A 0000               lbpl    wd_irq     yes! go
there!
         0022 B6   F010               lda     dma_c0     or is it
floppy dma?
       x 0025 102A 0000               lbpl    fd_irq     yes! go
there!

       x 0029 8E   0000      3        ldx     #n_chdv    then it
should be one
         002C A6   80                 lda     ,x+        of the
character devices.
         002E E6   02        4        ldb     q_mask,x   get IRQ
mask
         0030 E4   98 03              andb    [q_dadr,x] apply at
control register
         0033 27   05                 beq     5f         not this
device
         0035 EC   0B                 ldd     q_dev,x    get major
and minor
         0037 6E   98 09              jmp     [q_rout,x] jump to
IRQ routine/exit

         003A 30   00        5        leax    Lcdevirq,x skip this
device
         003C 4A                      deca               one less to
```

```
go
        0030 26   2F                bne    4b        next acia

        003F B6   F070              lda    fdc_00    INTRQ latch
on floppy board
      X 0042 102B 0000              lbmi   fd_irq    bit 7:
WD1791 INTRQ
      X 0046 1026 0000              lbne   wd_irq    not bit 7:
WD1002-5 INTRQ
        004A 39                     rts
```

LISTING I. Simplified IRQ routine.

```
                ttl     bootstrap for dmaf2/wd1002
               sttl    last modified 880715vbs
               pag
               *
               * TEMPORARY STORAGE $00E0-$00F6
               *       $00e0   1 of 4 used for divide routine
               *       $00e1   2 of 4 used for divide routine
               *       $00e2   3 of 4 used for divide routine
               *       $00e3   4 of 4 used for divide routine
               *       $00e4   1 of 2 used for divide routine
               *       $00e5   2 of 2 used for divide routine
               *       $00e6   1 of 2 used for divide routine
               *       $00e7   2 of 2 used for divide routine
               *
               *       $00e8   1 of 4 used for file date
               *       $00e9   2 of 4 used for file date
               *       $00ea   3 of 4 used for file date
               *       $00eb   4 of 4 used for file date
               *
               *       $00ec   1 of 1 used for number of blocks in list
               *       $00ed   1 of 2 used for pointer to block list
               *       $00ee   2 of 2 used for pointer to block list
               *       $00ef   1 of 3 used for block number
               *       $00f0   2 of 3 used for block number
               *       $00f1   3 of 3 used for block number
               *
               *       $00f2   1 of 2 used for number of entries in
directory
               *       $0023   2 of 2 used for number of entries in
directory
               *
               *       $00f4   1 of 2 used for start address
               *       $00f5   2 of 2 used for start address
               *
               *       $00f6   1 of 1 used for tandem head drives
               *
               *       hardware addresses
               *
      dma_al   equ     $f004   DMA transfer address channel 1
      dma_bl   equ     $f006   DMA byte count channel 1
      dma_cl   equ     $f011   control channel 1
      dma_pc   equ     $f014   priority control register
      dma_ic   equ     $f015   interrupt control register
               *
      extadr   equ     $f040   extended addressing latch
               *
      wd_dat   equ     $f060   WD1002-5 data register
      wd_e_v   equ     $f061   WD1002-5 register
      wd_cnt   equ     $f062   WD1002-5 count register
      wd_sec   equ     $f063   WD1002-5 sector register
      wd_cll   equ     $f064   WD1002-5 lsb cylinder register
      wd_clh   equ     $f065   WD1002-5 msb cylinder register
      wd_shd   equ     $f066   WD1002-5 side/head/drive register
      wd_s_c   equ     $f067   WD1002-5 status and control register
               *
               *       start of bootstrap program
               *
      wdboot   lbra    start   skip match string (14 byte)

      string   fcc     "uniflex"
               fcb     0,0,0,0,0,0,0

      start    ldx     #$b000  start by moving itself up $800 bytes
               ldu     #$b800  monitor should be modified...

      mvboot   ldd     0,x++
               std     0,u++
               cmpx    #$b200  all moved?
               blo     mvboot  no, move more.
               lbra    warmst+$800 go to new warmstart address

      warmst   ldd     #$0001  set fdn = 1, root ('/') directory
               lbsr    getfdn  read fdn 1
               ldd     7,y     get size of root (Y points to start of
fdn)
               bsr     div_16  16 bytes per entry
               std     <$00f2  store number of entries in root

      doroot   lbsr    getfil  read to buffer
               lbne    wd_ret  error exit

      donext   leax    2,u     skip fdn number
               leay    string,pcr point to 'uniflex'
               ldb     #14     match all characters

      wdcomp   lda     ,x+     match on this character?
               cmpa    ,y+
               bne     no_matc no, take next directory entry
               decb            ok, check next character
               bne     wdcomp  loop
               ldd     ,u      get fdn number (first 2 bytes of
dir.entry)
               bra     found   now look up fdn

      no_matc  ldx     <$00f2  'number of entries' counter
               leax    -1,x    one entry less
               stx     <$00f2  update counter
               beq     wd_ret  all entries tried, give up
               leau    16,u    next entry
               cmpu    #$bc00  end of buffer reached?
               bne     donext  no, loop
               bra     doroot  yes, read next part of directory into
buffer

      found    bsr     getfdn  look up fdn for 'uniflex'
               ldx     <$00ed  point to buffer
               ldd     39,x    take date out of fdn
               std     <$00e8  store
               ldd     41,x    take date out of fdn
               std     <$00ea  store
               bsr     getfil  go get the file
               bne     wd_ret  error, give up
               ldd     10,u    load start address
               std     <$00f4  store start address
               leau    24,u    start of first segment

      seginf   bsr     getbyt  get number of bytes this segment
               tfr     b,a
               bsr     getbyt
               pshs    a,b
               ldx     ,s++    number bytes now in X, stack cleaned up
               beq     loaded  if zero bytes to load, it's done
               bsr     getbyt  get address to load this segment
               tfr     b,a
               bsr     getbyt
               tfr     d,y     load address now in Y

      getseg   bsr     getbyt  load this segment
               stb     ,y+
               leax    -1,x
               bne     getseg  loop until done
               bra     seginf  get next segment information

      loaded   ldu     $5002   copy date from fdn into uniflex
               ldd     <$00e8
               std     ,u++
               ldd     <$00ea
               std     ,u++
               jmp     [$00f4] start uniflex

      div_16   lsra
               rorb
      div_8    lsra
               rorb
               lsra
               rorb
               lsra
               rorb
               rts

      getbyt   cmpu    #$bc00  check for end of buffer
               bne     ok      no, get byte in B
               pshs    a,x,y
               bsr     getfil  refill buffer
               puls    a,x,y
               bne     error   error exit

      ok       ldb     ,u+     get byte
               rts

      error    puls    a,b     clean up stack

      wd_ret   rts

      getfdn   pshs    a,b     save fdn number
               addd    #$000f  offset is 2 blocks/16 fdn's
               bsr     div_8   8 entries per block
               std     <$00f0  store 3 byte block number
               clr     <$00ef  msb always 0
               ldy     #$00ef  point to 3 byte block number
               bsr     readfd  read block into fdn buffer
               puls    a,b     recover fdn number
               bne     error   error exit
               decb
               andb    #7      8 FDN's per block
               lda     #$40    64 bytes per FDN
               mul
               addd    #$bc00  find start of fdn
               tfr     d,y     keep in Y
               addd    #$0009  pointer to block list
               std     <$00ed  save pointer
               lda     #$0a    10 blocks to go (direct blocks)
               sta     <$00ec  save number of blocks
               rts
```

```
getfil  tst     <$00ec  blocks available?
        beq     lb0113  get list of single indirect blocks
        dec     <$00ec  one less to read
        ldy     <$00ed  point to list of blocks
        ldx     #$ba00  file buffer pointer
        bsr     readbl  go read block in this buffer
        pshs    cc      save status
        sty     <$00ed  update block pointer
        puls    cc,pc   return

lb0113  ldy     <$00ed  point to first indirect block
        bsr     readfd  get it into buffer
        bne     lb00d1  errors
        stu     <$00ed  store pointer to next block
        lda     #$80    128 single indirect blocks
        sta     <$00ec  store number of blocks
        bra     getfil  start all over

readfd  ldx     #$bc00  buffer for fdn storage
readbl  leau    ,x      copy bufferaddress to U
        lbsr    setdma  set up DMA controller
        ldb     ,y+     get msb byte of block number
        clra            clear msb byte
        std     <$00e0  store for divide
        ldd     ,y++    get lsb bytes of block number
        std     <$00e2  store for divide
        ldd     sects,pcr       number of sectors
        std     <$00e4  store divisor
        bsr     divide  divide $00E0/3 by $00E4/5
        ldb     <$00e7  residu is sector number
        incb
        stb     wd_sec  put it sector we want in the WD1002
        ldd     heads,pcr       number of heads
        std     <$00e4  store as divisor
        bsr     divide  divide again to get the cylinder
        ldd     <$00e2  cylinder number
        clr     <$00f6  erase flag for tandem heads
        tst     cylflg,pcr      flag for high cyl. number
        beq     wd_cmd  no, put in WD1002 as is
        cmpd    cyls,pcr        cyl. OK?
        bcs     wd_cmd  yes, put in wd1002
        subd    cyls,pcr        no, subtract number of cyl.
        inc     <$00f6  set flag for tandem heads

wd_cmd  sta     wd_clh  put cylinder in WD1002
        stb     wd_cll
        ldb     <$00e7  get head (remainder of last division)
        orb     #%10100000      or-in ECC bit
        pshs    b       temp. save on stack
        ldb     <$00f6  get tandem head flag
        aslb            put in correct place
        aslb
        aslb
        orb     ,s+     place in shd template
        orb     $bfff   get drive from TurboBUG monitor
        stb     wd_shd  put in WD1002
        lda     #$28    read dma mode
        sta     wd_s_c  put in WD1002

wdwait  lda     dma_cl  check status on DMA channel
        bpl     wddend  wait for DEND flag
        lda     wd_s_c  check status of WD1002
        bpl     wdwai2
        bsr     wai500  wait some more...
        bra     wdwait

wdwai2  bita    #1
        beq     wdwait
        rts

wddend  ldd     #$ffff
        std     dma_pc  (and dma_lc) all quiet now
        clra
        clrb
        rts

divide  ldb     #$21    32 bit operation
        pshs    b
        clra
        clrb
        std     <$00e6
        bra     wdcalc

divid1  ldd     <$00e6
        subd    <$00e4
        bcs     wdcalc
        std     <$00e6

wdcalc  rol     <$00e3
        rol     <$00e2
        rol     <$00e1
        rol     <$00e0
        rol     <$00e7
        rol     <$00e6
        dec     ,s
        bne     divid1  loop over
        com     <$00e0
        com     <$00e1
        com     <$00e2
        com     <$00e3
        lsr     <$00e6
        ror     <$00e7
        puls    b,pc    clean up stack/return
setdma  lda     $010b   get fys. page at $0BXXX
        tfr     a,b     make copy
        lsra
        lsra
        lsra
        lsra
        ora     #$10
        sta     extadr  put in extended addr. latch on DMF-2
        eorb    #$ff
        aslb
        aslb
        aslb
        aslb
        pshs    b
        tfr     x,d
        anda    #$0f    max. is 4K transfer
        ora     ,s+
        coma
        comb
        std     dma_al  address to transfer to
        ldd     #!512   only nice blocks
        std     dma_bl  number of bytes
        lda     #%11111111 write in memory
        sta     dma_cl  put in channel control register
        lda     #%11111101 this channel only
        sta     dma_pc  put in priority control register
        rts
wai500  ldx     #500    wait loop
wdloop  leax    -1,x
        bne     wdloop
        rts

cyls    fdb     321

sects   fdb     17

heads   fdb     02

cylflg  fcb     0

filler  rzb     (512 - filler)

* end of boot sector
```

```
    1122                            *
    1123                            * overwrite root, pipe en swap device
    1124                            * in UniFLEX, as set up by TurboBUG
    1125                            *
    1126 *                 0338  init02    equ    *
    1127   0338 9E   32                    ldx    rootdev
    1128 x 033D BF   0000                  stx    rootdv    overwrite
default
    1129
    1130   0340 9E   34                    ldx    pipedev
    1131 x 0342 BF   0000                  stx    pipedv    overwrite
default
    1132
    1133   0345 9E   36                    ldx    swapdev
    1134 x 0347 BF   0000                  stx    swapdv    overwrite
default
    1135
    1136                            *
    1137                            * clear $0019-$00e0
    1138                            *
```

LISTING 3. Part of initialization.

Bit-Bucket

By: All of us

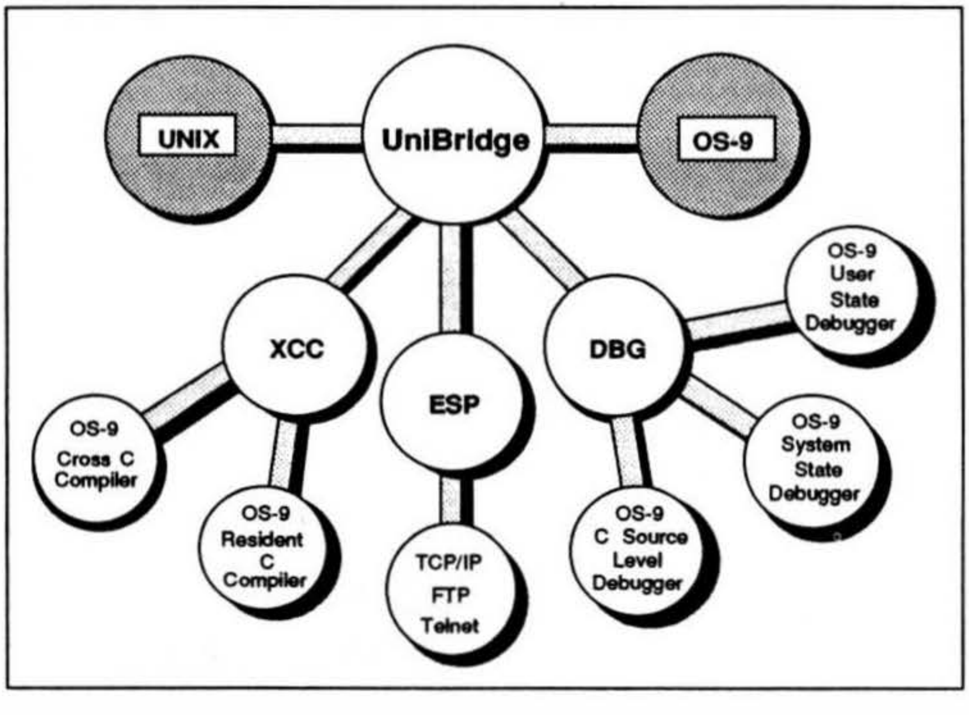
"Contribute Nothing - Expect Nothing", DMW '86

OS-9 UniBridge Connects UNIX to Real Time

UniBridge is an advanced software package for C language development and Ethernet communication that connects UNIX to OS-9. With UniBridge VMEbus system integrators and designers can now develop real-time applications using popular UNIX-based workstations. This allows OS-9 systems to be used with popular SUN, DEC VAX, HP and Motorola UNIX workstations for distributed development and real-time supervisory control.

Software engineers can use UniBridge to connect the rich development environment of UNIX to the powerful real-time capabilities of OS-9. The UniBridge package contains all of the sophisticated tools needed to make the connection between a UNIX host and OS-9 target. UniBridge includes:

OS-9/XCC UniBridge contains both UNIX and OS-9 resident C Compilers for 68000 or 68020 microprocessors with full 68881 support. Users can compile on the host or target to produce compact, re-entrant, position independent object code for real-time execution. Since the OS-9 C Compiler utilizes UNIX C standard libraries, C programs can be compiled with OS-9/XCC to operate on OS-9 resident systems without program conversion. UNIX standard libraries also allow OS-9 C programs to be easily ported to the host environment.

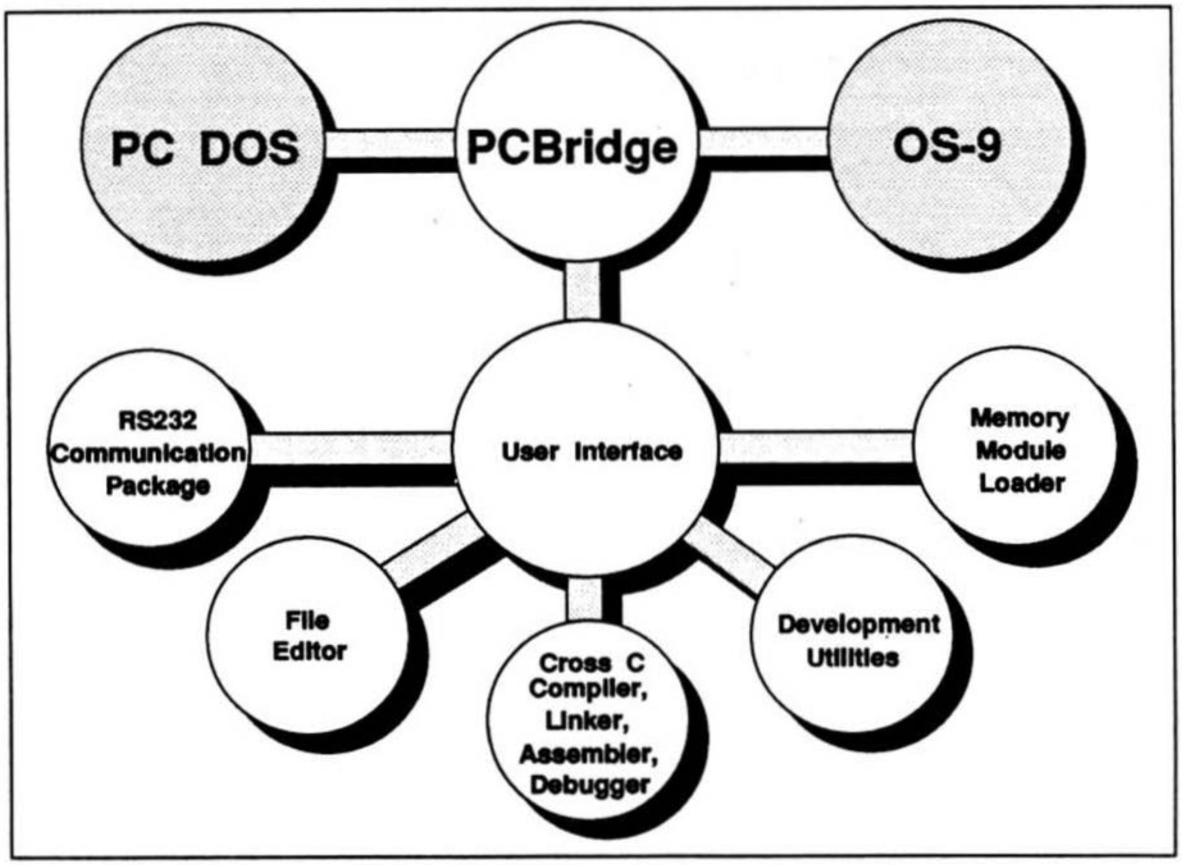


UniBridge Modules.

PCBridge Links PC's to Real Time

PCBridge is an easy-to-use PC-hosted development system for OS9/680x0 applications. Through PCBridge, MS-DOS users gain access to the OS-9 Operating System. PCBridge provides a C cross compiler, assembler and linker, and a set of program development utilities. These utilities include terminal emulation, text and binary file transfers between MS-DOS and OS-9, file manipulation utilities and session logging. The development utilities are distributed between the host and target systems.

PCBridge provides a platform for distributed applications, building synergistically upon the real-time aspects of OS-9 and an easy-to-use PCBridge interface under MS-DOS. For example, OS-9 can be used for such tasks as real-time data acquisition, image processing, factory and robotics control. PCBridge can link these types of systems for process monitoring, status and control to information management applications on PC-DOS. A total distributed application can be developed and integrated using PCBridge. The system uses a front-end PC/XT/AT which is linked to the target OS-9 system via a high-speed serial line. The user interface is pop up menu-driven, with the user selecting a function (Edit, Compile program, load memory module, etc.) indicatingnecessary parameters, and the PC and OS-9 systems cooperate to perform the operation without further user intervention. Selections are made via keyword, keypad cursor keys, or a mouse.



PCBridge Modules

The target OS-9 system can be almost any configuration, including ROM-based with limited RAM and no disk. Applications can be developed on the PC and loaded into the target system across a high-speed serial line for execution or testing.

PCBridge is distributed on either 3 1/2" or 5 1/4" diskettes, and come with complete professional documentation and free 90-days "Hotline" support. For additional information on PCBridge, contact Microware today.

OS-9/ESP The OS-9/ESP Ethernet Support Package provides complete Ethernet TCP/IP communication between the host environment and OS-9 based systems. OS-9/ESP incorporates both FTP and Telnet protocols for efficient file transfer and remote login capabilities. Users can easily access OS-9 from UNIX, UNIX from OS-9 and OS-9 from OS-9 for distributed software development and supervisory real-time execution. OS-9/ESP features a C compatible Berkeley 4.2 socket library combined into an Internet database as a single OS-9 data module.

OS-9/SRCDBG OS-9/SRCDBG combines both a full-featured C Source Level Debugger and System State Debugger to provide a rich environment for testing and debugging OS-9 C Language programs. The C Source level Debugger features a C expression interpreter and an extensive command set which allows the user to debug any OS-9 C program at the source level. Users have the ability to invoke debugger control and communication, data manipulation and system commands to significantly decrease software development time.

UniBridge comes with complete professional documentation and free 90-day "Hotline" support. For additional information on UniBridge, contact Microware today .

MICROWARE UPDATE POLICY

Microware offers customers who purchased Version 1.0 of any end-user software product an update to the next version free of charge. Contact Microware within 90-days of a new version's release for complete update information.

C SOURCE LEVEL DEBUGGER

A new release of the Source Level Debugger is now available. Version 2.0 has been optimized for even higher performance and includes many new powerful features. These features include: Debugging multiple module programs (i.e. trap handlers and subroutine modules); Assembly level debugging; Complete access to processor registers in C Language expressions; conditional break points, break counts and command scripts; and complete stack backtrace capabilities. All modifications made in Version 2.0 have been fully documented in release notes
included with the updated software.

The complete list of Version 2.0 C Source Level Debugger Commands:

asm(.)	a[ssign]	b[reak]	c[hange]
c[h]c	c[h]d	c[h]x	con[text]
di[asm]	dil[ist]	d[ump]	fi[nd]
fo[rk]	f[rame]	g[o]	gostop(gs)
i[nfo]	k[ill]	li[nk]	l[ist]
lo[cals]	l[o]g	mf[ill]	ms[earch]
n[ext]	o[ption]	p[rint]	re[ad]
r[eturn]	se[tenv]	shell($)	s[tep]
sy[mbol]	t[race]	unse[tenv]	w[atch]

New C Source Level Debugger Commands added to Version 2.0:

b[reak] [<location_expr>] [:wh[en] <C_expr>]
[:co[unt] <num>]

The user may set conditional break points, break counts and breaks at source or assembly language locations.

c[h]c [<scope_expression>]

This enables the user to set break points etc. without the use of scope/line number expressions.

c[h]x <pathlist>

Changes the current execution directory for SrcDbg.

con[text] [<scope_expression>]

Fully qualifies a symbol in terms of scope. Informs the user exactly which symbol is going to be referenced in an expression.

fi[nd] [<name>]

Displays all scope expressions found for <name>. Informs user of all occurrences of a name.

f[rame] [[+ | -] <number>]

Changes stack frame to <number>. Frame with no arguments displays current call stack frame information. User now can access local variables in the function call stack.

g[o] [<location_expr>] [:dis[play]]

Go now provides a way to run the program to a certain spot without the user setting and removing a break point.

l[o]g <pathlist> | : off

Writes SrcDbg commands to <pathlist>. ": off", closes the log file. The user can now save a series of commands and re-execute them at a later time.

lo[cals]

Displays the values of all local symbols. This provides a quick way of referencing local variables.

o[ption] { <options>}

Options:

fpu	toggle fpu register display.
fregs	toggle fpu display between hex and decimal.
rom	toggle rom (soft) and ram (hard) breakpoints.
source	toggle source display during assembly level displays.
watch	toggle location display after watch expression changes.
dbg	toggle reading of ".dbg" files.
stb	toggle reading of ".stb" files.
prompt	toggle prompt output.
echo	toggle command line output.

These provide the user with greater control of the source debugger and its displays.

re[ad] [<pathlist>]
Reads SrcDbg commands from <pathlist> and enables the debugger to read command scripts. These may be created by the user with an editor or with the "log" command.

se[tenv] <environment_name>
<environment_definition>
Sets a shell type environment variable. The user can now change the environment for the use of the debugged program or the debuggers' environment itself.

sy[mbol] [<C_expr>]
Displays the result of the expression as a symbolic expression. This command is useful in showing what symbol a pointer variable is referencing.

unse[tenv] <environment_name>
Deletes environment variable. Provides further control
over the environment.

New Assembly Level Commands:

c[hange] [<C_expr>]
This command provides an easy way to change byte(s), word(s), or longword(s) in memory.

gostop | gs[<number>]
Executes <number> of machine instructions in the current subroutine. Similar to the "next" command but at assembly level.

li[nk] <module_name>
Links to <module_name> and places module address in ".r7". A user can load/link a memory module and then use ".r7" in C Language expressions to access it.

dil[ist] [<location_expr>][: [<count>]]
Displays C source with disassembly. This gives the user a way to see the assembly code that is mapped to their C language code.

di[sasm] [[<C_expr>] [: [<count>]]]
Disassembles memory at the result of <C_expr>. Provides a means to display assembly code.

d[ump] [[<C_expr>] [: [<count>]
[<format>]]]
Displays memory at the result of <C_expr>. This formatted memory dump command has user controls on the display format.

mf[ill] <begin> : <end> : <value>
Fills memory with <value>. This command provides an easy way to fill memory with a desired bit pattern.

ms[earch] <begin> : <end> : <value>
[: <mask>]
Searches memory for <value>. Provides an easy way to search memory for a desired bit pattern.

t[race] [<number>]
Provides a way to step through assembly language code an instruction at a time.

OS-9/68000 FORTRAN 77 COMPILER

VERSION 1.2
Microware has released FORTRAN 77, Version 1.2. This new edition update includes corrections for known problems in the "fort" executive and both compiler phases, "fortp1" and "fortp2". In addition, the fortp2 user error messages are now displayed with more meaningful descriptions. For example the error message "can't write temporary file" will be displayed when -t=/r0 is used and the RAM disk fills up. The D floating point notation format is now also supported. All modifications made in Version 1.2 have been fully documented in release notes included with the updated software. The Fortran update is available for both 68000 and 68020 target systems.

ETHERNET SUPPORT PACKAGE

(ESP) - VERSION 1.1
Microware announces the release of ESP Version 1.1. This new edition update incorporates bug fixes and enhancements to improve the reliability and performance of the ESP software.

Two new header files appear in the DEFS directory. errno.h is identical to the errno.h supplied with the 3.0 C Compiler. It includes the error number definitions for the socket errors. sgstat.h is the get/setstat struct file and contains additional definitions for the get/setstat options call to the ENP10 driver.

The ETC directory includes a new error message file "errmsg.short" which incorporates the error messages, mentioned in the errno.h description, into a file to use as the "/dd/sys/errmsg" file on the OS-9 target system.

A file "enp.gate" describes how to change the "enp.a" device descriptor to direct packets to a gateway machine for internet routing. This can be found in the ENP directory.

The following hardware and software is minimally require to install and run OS-9/ESP:
. ENP-10+ Board with V4.1 K1 Kernel ROMs
. OS-9 System running V2.2 or later
. Ethernet LAN system

For complete list of all changes made in this edition update, please refer to the full release notes provided on the shipment floppy.

+++

DO A FLEX-09 Batch File Processor

By: Dave Howland

Continued from page 54 of Nov. 1988 issue.

```
ECHO OFF
!
! DEL.BAT
! Batch file to delete multiple files
! Written by Dave Howland
! Vesion 1.1, 27th December 1987
!
! If no parameters or ‘+h’, output help info
!
IF ‘%1 = ‘ GOTO help
IF %1 = +h GOTO help
!
! Perform loop until all files processed
!
IF ‘%1 = ‘ GOTO help
@loop
  IF EXIST %1 THEN
    delete %1
    Y%c
  ELSE
    NOTE %1 doesn’t exist
  ENDIF
  SHIFT
IF NOT ‘%1 = ‘ GOTO loop
GOTO exit
!
! Output help info
!
@help
NOTE Usage : DO DEL <file> <file> ...
!
! Single exit point
!
@exit
```

```
ECHO OFF
!
! DEV.BAT
! Batch file for development of assembler programs
! Written by Dave Howland
! Version 1.1, 21st December 1987
!
! If no parameters or ‘+h’, output help info
!
IF ‘%1 = ‘ GOTO help
IF %1 = +h GOTO help
!
! Perform loop until user does not wish to
! continue the development cycle
!
@loop
  edit %1.txt
  %t
  NOTE Assemble source file
  IF %c = y THEN
    IF EXIST %1.cmd THEN
      delete %1.cmd
      yy
    ENDIF
    asmb %1.txt %1.cmd +ls
  ENDIF
  NOTE Continue development cycle
IF %c = y GOTO loop
GOTO exit
!
! Output help info
!
@help
NOTE Usage : do dev <file>
!
! Single exit point
!
@exit
```

```
    *
    * Header file for FLEX programs, version 1.3, 9th September 1987
    *
    * FLEX memory map
    *
    LINBUF  EQU     $C080           command line input buffer (128)
    CMDADR  EQU     $C100           utility command space
    CMDEND  EQU     $C6FF           utility command space end
    SYSFCB  EQU     $C840           system FCB (320)
    TTYBS   EQU     $CC00           TTYSET backspace char
    TTYDEL  EQU     $CC01           TTYSET delete char
    TTYEOL  EQU     $CC02           TTYSET end of line char
    TTYDEP  EQU     $CC03           TTYSET page depth count
    TTYWID  EQU     $CC04           TTYSET page width count
    TTYNUL  EQU     $CC05           TTYSET null count
    TTYTAB  EQU     $CC06           TTYSET tab char
    TTYBSE  EQU     $CC07           TTYSET backspace echo char
    TTYEJ   EQU     $CC08           TTYSET eject count
    TTYPS   EQU     $CC09           TTYSET pause control (0 = enabled)
    TTYESC  EQU     $CC0A           TTYSET escape char
    SYSDRV  EQU     $CC0B           system drive number
    WRKDRV  EQU     $CC0C           work drive number
    SYSFLG  EQU     $CC0D           use system drive flag
    DATE_M  EQU     $CC0E           system date - months
    DATE_D  EQU     $CC0F           system date - days
    DATE_Y  EQU     $CC10           system date - years
    LSTTRM  EQU     $CC11           last terminator, after NXTCH or
CLASS calls
    USRCMD  EQU     $CC12           pointer to user command table (2)
    LINPTR  EQU     $CC14           pointer to next char in LINBUF (2)
    ESCRET  EQU     $CC16           escape return address (set to WARMS)
(2)
    CURCHR  EQU     $CC18           current char returned by NXTCH
    PRVCHR  EQU     $CC19           previous char returned by NXTCH
    CURLIN  EQU     $CC1A           current line number on a page
    LDOFF   EQU     $CC1B           loader offset address (2)
    TFRFLG  EQU     $CC1D           transfer address found while loading
(0 = no)
    TFRADR  EQU     $CC1E           transfer address, if TFRFLG = yes
(2)
    ERRTYP  EQU     $CC20           error code returned by FMS
    SPECIO  EQU     $CC21           ignore TTYSET width and escape (0 =
no)
    OUTSWT  EQU     $CC22           PUTCHR output switch (0 = OUTCH, $ff
= OUTCH2)
    INPSWT  EQU     $CC23           GETCHR input switch (0 = INCH, $ff =
INCH2)
    FILOUT  EQU     $CC24           address of FCB for file input via
GETCHR (2)
    FILIN   EQU     $CC26           address of FCB for file output via
PUTCHR (2)
    CMDFLG  EQU     $CC28           DOCMND flag (0 = not, $ff = called
via DOCMND)
```

```
    CURCOL  EQU     $CC29           current column number in line
    MEMEND  EQU     $CC2B           end of user memory (2)
    ERRVEC  EQU     $CC2D           address of errors filename (0 =
ERRORS.SYS) (2)
    FILECH  EQU     $CC2F           file input echo flag (0 = no echo)
    ULCFLG  EQU     $CC49           case flag ($60 = lower -> upper, $ff
= not)
    PROMPT  EQU     $CC4E           prompt string
    *
    *
    * Printer routines
    *
    PRTINT  EQU     $CCC0           printer initialise routine
    PRTCHK  EQU     $CCD8           printer status check routine
    PRTOUT  EQU     $CCE4           printer output routine
    PRTBSY  EQU     $CCFC           print spooler busy status (0 = not
busy)
    *
    *
    * System routines
    *
    COLDS   EQU     $CD00           cold start entry point
    WARMS   EQU     $CD03           warm start entry point
    RENTER  EQU     $CD06           dos main loop re-entry point
    INCH    EQU     $CD09           input char to ACCA (alterable)
    INCH2   EQU     $CD0C           input char to ACCA (not alterable)
    OUTCH   EQU     $CD0F           output char from ACCA (alterable)
    OUTCH2  EQU     $CD12           output char from ACCA (not alter-
able)
    GETCHR  EQU     $CD15           get char into ACCA (uses INCH or
INCH2)
    PUTCHR  EQU     $CD18           put char from ACCA (uses PUTCH or
PUTCH2)
    INBUFF  EQU     $CD1B           input line to LINBUF, reset LINPTR
    PSTRNG  EQU     $CD1E           print CR/LF and IX -> string
    CLASS   EQU     $CD21           classify char in ACCA (zero carry =
alphanum)
    PCRLF   EQU     $CD24           print CR/LF
    NXTCH   EQU     $CD27           get next char from LINBUF, exit via
CLASS
    RSTRIO  EQU     $CD2A           restore IO vectors (xxCH = xxCH2,
xxSWT = 0)
    GETFIL  EQU     $CD2D           get file spec from LINBUF to IX ->
FCB
    LOAD    EQU     $CD30           load file, name in SYSFCB
    SETEXT  EQU     $CD33           set extension in IX -> FCB, code
(below) in ACCA
    ADDBX   EQU     $CD36           add ACCB to IX
    OUTDEC  EQU     $CD39           decimal output, IX -> 2 byte value
    OUTHEX  EQU     $CD3C           hex output, IX -> 1 byte value
    RPTERR  EQU     $CD3F           report error, IX -> FCB containing
error code
    GETHEX  EQU     $CD42           get hex number from LINBUF into IX
    OUTADR  EQU     $CD45           hex output, IX -> 2 byte value
    INDEC    EQU     $CD48          get decimal number from LINBUF into
IX
    DOCMND   EQU     $CD4B          call DOS, command in LINBUF, LINPTR
-> command
    STAT     EQU     $CD4E          check terminal input status (Z clear
= char)
    *
    *
    * File management system - entry points
    *
    FMSINT   EQU     $D400          initialise FMS
    FMSCLS   EQU     $D403          close all open files
    FMS      EQU     $D406          call FMS, IX -> FCB
    *
    *
    * File management system - global variables
    *
    FMSBAS   EQU     $D409          pointer to FCB chain, or 0 (->
F_CPTR) (2)
    FMSCUR   EQU     $D40B          pointer to last processed FCB (->
F_FUNC) (2)
    FMSVER   EQU     $D435          read after write verify flag (0 =
no)
    *
    *
    * File management system - function codes
    *
    M_READ   EQU     0              read next byte from file open for
read
    M_WRIT   EQU     0              write next byte to file open for
write
    M_OPNR   EQU     1              open file for read
    M_OPNW   EQU     2              open file for write
    M_OPNU   EQU     3              open file for update
    M_CLOS   EQU     4              close file
    M_RWND   EQU     5              rewind file opened for read
    M_OPND   EQU     6              open directory
    M_GETD   EQU     7              get directory information record
    M_PUTD   EQU     8              put directory information record
    M_RSEC   EQU     9              read single sector
    M_WSEC   EQU     10             write single sector
    M_DEL    EQU     12             delete file
    M_REN    EQU     13             rename file (new name in F_SCR)
    M_NEXT   EQU     15             next sequential sector
    M_OPNS   EQU     16             open system information record
    M_GETR   EQU     17             get random byte from sector
    M_PUTR   EQU     18             put random byte to sector
    M_NXTD   EQU     20             find next drive
    M_MOVR   EQU     21             move to any record in random file
    M_PRVR   EQU     22             move to previous record in random
file
    *
    *
    * File Control Block offsets
    *
    F_FUNC   EQU     0              function code
    F_ERR    EQU     1              error status byte
    F_ACT    EQU     2              activity status (1 = read, 2 =
write)
    F_DRIV   EQU     3              drive number (0 to 3)
    F_NAME   EQU     4              file name (left just., filled with
0) (8)
    F_EXT    EQU     12             file extension (left just., filled
with 0) (3)
    F_ATTR   EQU     15             file attributes (1 = set)
    F_STRK   EQU     17             starting track/sector of file (2)
    F_ETRK   EQU     19             ending track/sector of file (2)
    F_SIZE   EQU     21             number of sectors in file (2)
    F_FSM    EQU     23             file sector map flag (0 = seq, $02 =
random)
    F_DATE   EQU     25             file creation data (month, day,
year) (3)
    F_CPTR   EQU     28             FCB chain pointer (2)
    F_TRKS   EQU     30             track/sector of sector in F_SBUF (2)
    F_REC    EQU     32             logical record number of sector in
F_SBUF (2)
    F_INDX   EQU     34             offset of next data byte in F_SBUF
    F_RND    EQU     35             offset of random data byte to access
in F_SBUF
    F_DIR    EQU     47             track/sector/offset of dir entry in
FCB (3)
    F_SCR    EQU     53             new name/extension of file being
renamed (11)
    F_SCF    EQU     59             space compression flag (0 = yes)
    F_SBUF   EQU     64             sector buffer (bytes 0-3 = link, 4-
255 = data)
    F_LEN    EQU     320            FCB length
    *
    *
    * File protection code masks
    *
    P_CAT    EQU     $10            catalogue protected
    P_READ   EQU     $20            read protected
    P_DEL    EQU     $40            delete protected
    P_WRIT   EQU     $80            write protected
    *
    *
    * System Information Record offsets
    *
    S_VNAM   EQU     16             volume name
    S_DTKS   EQU     24             current directory track/sector
    S_VNUM   EQU     27             volume number
    S_FRST   EQU     29             first track/sector in free chain
    S_FRND   EQU     31             last track/sector in free chain
    S_FRLN   EQU     33             number of sectors in free chain
    S_DATE   EQU     35             disk creation date
    S_HTRK   EQU     38             highest track number on disk
    S_HSCT   EQU     39             bighest sector number on disk
```

To be Continued Next Month

MOTOROLA INC.

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-6598

MOTOROLA DSP PROCESSOR PROVIDES KEY FEATURES IN NeXT COMPUTER SYSTEM

NeXT Uses 56001 For CD-Quality Sound, Speech Synthesis, Modem, FAX

Alan Kelly
Cunningham Communication, Inc.
(408) 982-0400

Jane Bates
Microprocessor Products Group
(512) 440-2033

AUSTIN, Texas, Oct. 12, 1988 — Motorola's Microprocessor Products Group today announced that its high-performance digital signal processor (DSP), the DSP56001, will power revolutionary features offered in the new computer from NeXT, Inc. (Palo Alto, Calif.). The 56001 gives the NeXT™ Computer System key capabilities in compact disc quality sound, speech synthesis, a high-speed modem, facsimile transmission, array processing, voice mail and high-speed numeric processing. NeXT is the first computer manufacturer to integrate the 56001 in all its systems.

"Our system completely redefines how people will interact with computers," said Steve Jobs, president and chairman of NeXT. "The 56001 helps us provide in a single system an incredible depth of applications from symphony-like sound to a high-speed modem."

Introduced in March 1987, the Motorola 56001 is a general purpose digital signal processor whose architecture is optimized for high data throughput and real time processing. It is designed directly onto the motherboard of the NeXT Computer System, and operates in conjunction with Motorola's 68030 and 68882 central processing engines.

The 56001 provides the basis for on-board data communications (e.g. fax and modem) and sound synthesis (e.g. voice mail, voice interactive programs, sound editing and high-fidelity audio). It recreates CD-quality sound because its architecture offers high data throughput and 144 decibels of dynamic range. The processor also modulates and demodulates data signals as part of the machine's high-speed internal modem. With its speech synthesis capabilities, it allows the NeXT system to have integrated voice mail.

"We see the NeXT system as a key endorsement of our DSP technology," said Bryant Wilder, operations manager of DSP for Motorola. "The 56001 gives this product tremendous value and lays the groundwork for some very exciting applications."

Simplified Application Development

The architectures of both the 56001 and the NeXT Computer System give users and programmers complete freedom to create digital signal processing applications.

In the NeXT system, all of the 56001's on-chip peripherals and resources are made fully accessible to the system user. Conventional computer makers usually design peripheral chips into a closed environment, which causes users to depend solely on the vendor for application solutions.

The NeXT system, however, gives programmers total access to the 56001 to create innovative voice, sound and data communication applications in an unencumbered development environment. The 56001 also embodies a more flexible architecture that is better suited to general purpose programming, which helps speed the development process and makes DSP technology available to a broader community of programmers.

Also available to the user through Motorola are a full complement of third-party application development systems including: in-circuit emulation for debugging; fully functional development and system boards; a wide variety of software development tools including assemblers, simulators, a C compiler, applications notes and DSP routine libraries that are all designed to aid computer users who are unfamiliar with DSP.

24-Bit Architecture

The 56001 brings a key design advance to digital signal processing: 24-bit architecture. While most DSP chips process information in 16-bit word lengths, the 56001 is the only fixed-point DSP with 24-bit architecture. With the additional eight bits, the 56001 substantially boosts performance and tackles an unprecedented number of DSP applications. As a 24-bit chip, the 56001 handles the processing overflow created by 16-bit analog-to-digital converters and thus maintains high arithmetic precision.

The 56001 has been benchmarked as the fastest fixed-point DSP chip on the market. It processes a command in 97.5 nanoseconds, the time it takes for sunlight to move 96 feet. It operates at 10.25 MIPS (million instructions per second) at a 20.5 MHz clock rate.

MOTOROLA 68030 POWERS NEW NeXT COMPUTER

68882 Used as Math Coprocessor

Zachary Nelson
Cunningham Communication, Inc.
(408) 982-0400

Dean Mosley
Microprocessor Products Group
(512) 440-2839

AUSTIN, Texas, Oct. 12, 1988 — Motorola's Microprocessor Products Group today announced that its top of the line 68030 (030) and 68882 (882) microprocessors will power a new computer from NeXT, Inc. (Palo Alto, Calif.). The new machine, called the NeXT™ Computer System, uses Motorola's 25 MHz 030 as the central processing engine responsible for all information processing. The computer also uses a 25 MHz 882 processor to perform mathematical computations, an essential feature for scientific, engineering and business applications.

The 68000 product line is the leading microprocessor solution for high-performance, leading-edge systems. The 030 ("oh thirty") microprocessor is a fully compatible member of Motorola's 68000 family, which includes the 68000 and 68020 chips. The 68000 line is supported by over $100 billion in hardware and $3 billion in software, the world's largest base of 32-bit applications.

"Any computer manufacturer building an innovative, flexible system would choose Motorola's 68000 family as the foundation for the technology," said Steve Jobs, president and chairman of NeXT.

Since its introduction in 1979, the 68000 family has been the driving force behind the scientific and engineering workstation market, and it is widely credited for spawning the graphics revolution. Its general purpose register set, flexible architecture and ease of programming make it the choice for those companies designing user-friendly, intuitive interfaces.

"The history of the 68000 has been one of innovation, both by our customers and by our designers," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group. "The new NeXT system continues this legacy."

The 030 includes many features that increase the number of functions it can perform simultaneously and the rate at which it can feed information to its central execution unit. It is the first general-purpose microprocessor with on-chip cache memory for computer instructions and data. The 030 is also the first chip with an internal parallel architecture called Harvard-style architecture. With two independent address buses and two independent 32-bit data buses, the processor can access and use multiple data sources simultaneously.

Floating-point coprocessors are used to speed mathematical calculations in a variety of business, financial and engineering applications. The 882 floating-point math coprocessor provides sophisticated numeric processing functions on a single chip. It can execute instructions simultaneously with the 030 central processor, thereby increasing overall system performance. The 882 manipulates data as large as 80 bits (digits) long for increased accuracy.

NeXT, Inc., of Palo Alto, Calif., was founded in October 1985 by Steve Jobs, co-founder and former chairman of Apple Computer Inc., and five other individuals. The mission of the privately-held company is to collaborate with higher education to develop innovative, personal and affordable computer solutions for the 1990s and beyond.

Motorola's $2.2 billion Semiconductor Products Group Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a division of Motorola, Inc. The company is the largest and broadest supplier of semiconductors in North America, with a balanced portfolio of more than 50,000 devices.

MICRONICS
RESEARCH CORP.

Microcomputers - Hardware and Software
GIMIX® Sales, Service and Support

33383 LYNN AVENUE,
ABBOTSFORD,
BRITISH COLUMBIA,
CANADA, V2S 1E2

Dear Don,

To clear up any possible confusion in readers' minds regarding Symmetric Functions in my series "Logically Speaking", the notation didn't come out correctly in the October issue, and will probably also be incorrect for subsequent articles. First, any 'peculiar' asterisks in these lessons should be deleted by the reader - they are a "flag" set by me to indicate to the typesetter that a Symmetric Notation occurs in this line, and my 'on-disk' S6/1,2,4,5 ABCDEF, for instance, should actually be printed as

$$S^{6}_{1,2,4,5}\ ABCDEF$$

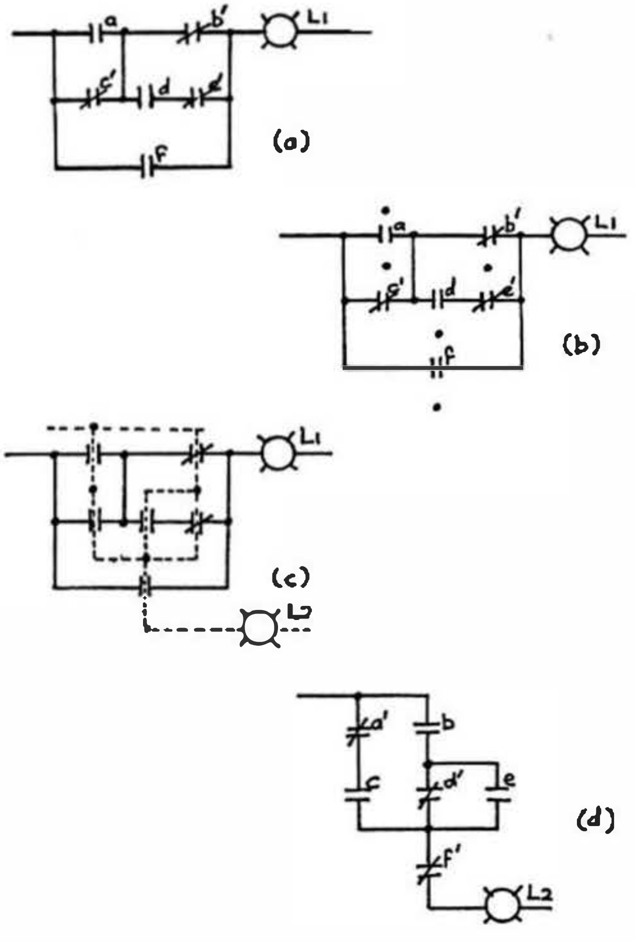
In addition, where only a partial conversion appears and you see something like

$$S^{6}_{1,}2,4,5\ ABCDEF$$

this should also appear as in the first example.

Page 36, para 3, sentence 3 should end with a '?', thus "... do NOT equal 1?"
Page 40, penult para - somehow or other a superfluous 'a' crept into the word 'horizontally'.

Page 57, and my letter of earlier corrections. Readers still won't know what Diagram 53 looks like, so I'll reproduce it here as part of this letter.



Klockargraend 10
S-191 72 Sollentuna
Sweden
October 24, 1988

Dear Don,

I recently managed to lay my hands on a 5-year-old, second-hand UNIX machine. However, the seller warned me that: a) it was the only one in Sweden, b) the manufacturer went down the tubes a few years ago and c) they had unwittingly disposed of some of the documentation a few months earlier.

The machine is a VICTORY "Vector series", with 68000 processor, VME bus, 40 MB Winchester (made by Quantum) and a 5 MB cartridge Winchester (made by DMA). It runs UniPlus+ (UNIX System III).

Specifically, what I need is:
- the following sections, from the UniPlus+ version of "UNIX Vol. II, Program Development Tools" (or equivalent info):
    PART 2: Program Maintenance
     2.4.1 Source Code Control System User's Guide
     2.4.2 Function and Use of an SCCS Interface Program
    PART 3: UNIX Maintenance and Information
     3.1 to 3.6 (i.e. the whole of PART 3)
    PART 4: Networking
     4.1 to 4.3 (i.e. the whole of PART 4)
- any hardware/programming information available on the DMA drive (5 MB cartridge Winchester),
- any hardware/programming information available on the Quantum drive (40 MB Winchester),
- anything at all on the UniPlus+ implementation, for the VICTORY,
- a handbook for SVS BASIC+ (Silicon Valley Software),
- a source of reasonably-priced cartridges for the DMA drive (used, undamaged cartridges are OK),
- if anyone knows of Uniplus+ manuals with Vol. No. IV or higher, I would be very grateful for a copy of the 'Contents' pages.

In the event that anyone has access to any of the above information, please write to me (stating prices, where appropriate), at the above address. I promise to reply to all (polite) letters.

Yours sincerely,
Jason King
(Jason King)

d.p. johnson

microcomputer consulting

7655 southeast cedartree street • portland, oregon 97223 • (503) 244-8152

NEW PRODUCT RELEASE

FORTH09

D.P.Johnson announces the release of FORTH09, a Forth-83 language for use with OS-9 (level 1 or level 2 6809). FORTH09 code compiles to machine code. An application program in Forth can be saved as an OS-9 executable module which is inherently reentrant and relocatable. FORTH09 contains all of the Forth-83 required word set and all defined extensions including a full assembler. Many system call words are provided to make full use of the power of OS-9. A screen editor is built in with variants for several video terminals and CoCo-3 80 column screen. FORTH09 runs as any other OS-9 process, and uses the OS-9 file structure for its mass storage requirements. The user may force short code words to be automatically compiled as inline code for greater speed when desired.

FORTH09 is available immediately for $150.00 + $3.00 shipping ($10.00 for overseas airmail). The user manual may be purchased separately for $25.00 (+shipping) with the price applied toward later purchase of the software. Specify diskette size and format when ordering. Contact Dan Johnson for further information.

WINDRUSH
Micro Systems Ltd

Station Road (Park Office)
Worstead North Walsham
Norfolk NR28 9SA England
Telex 975648 WINCRO G
Tel (0692) 404086
Fax (0692) 404091

PRESS RELEASE

Ωmega Wins Gold in Price - Performance Olympics

Windrush Micro Systems Limited are pleased to announce the immediate availability of their Ωmega-II system which now includes OS-9/68020 'Professional' and the associated optimized MC68020/MC68881 C compiler.

Recent benchmarks carried out at Imperial College Computer Center found the Ωmega running 4194 Dhrystones per second thus comfortably outperforming the Sun 3/75, 3/160 and 3/180, IBM PC/AT and the Apple MAC II. The Fibonacci, Float, Savage, Sieve and Sort benchmarks also showed the Ωmega-II a clear leader.

The basic system incorporates a 12.5 MHz MC68020, 512K of zero wait-state static RAM, five RS-232C serial ports and a parallel printer port. This system is supplied with a single 1 mb 3.5" floppy disk and SCSI interface for a user supplied hard disk and OS-9/68020 Professional for £1,895 (1 off).

The top of the line Ωmega workstation, costing £4950 (1 off) incorporates a 16 MHz MC68020 processor and MC68881 math co-processor as standard and includes five RS-232 ports, a 40 Mb Winchester hard disc with a seek time of less than 30 mS, a 1 Mb 3.5 inch floppy disk, a 150 Mb 1/4" tape streamer, 2 megabytes of zero wait-state, non-volatile Static RAM. A parallel printer port, a clock calendar and OS-9/68020 Professional are also included.

For further information contact Bill Dickinson at (0692) 404086

P R O D U C T I N F O R M A T I O N

New Software by MSB for OS-9/68000

PROGRAMMABLE SHELL (UNIX/Bourne-shell) FOR OS-9/68000
MSB-SH

With the software package MSB-SH a powerful programmable Shell as a command interpreter is available. It is full compatible to the standard UNIX-shell (Bourne-shell) from UNIX V (BSD 4.2). Like the Microware-Shell, the MSB-SH is an interface to the OS-9/680XX operating system and sees itself as a command programming language. The features are parameter passing, variables, string substitution and control-flow primitives. Constructs such as 'while', 'if elif else', 'case', 'for' are available. Bi-directional communication between the Shell and commands is possible. String parameters (i.e. file names or flags) may be passed to a command. A return code is set by commands that may be used to determine control flow, and the standard output from a command may be used as Shell input.
The Shell can modify the environment in which commands run and can use it also by itself. Input and output can be redirected to files and processes that communicate through 'pipes' can be invoked.
Commands can be read either from the terminal or from a file, which allows command procedures to be stored for later use. These command procedures can be called with its own parameters.
With additional commands like 'test' and 'expr' it is possible to do calculations and check files and Shell variables for its kind and state.

Beyond the basic controll mechanisms for command grouping '&&' (AND) and '||' (OR), the following language elements are provided:

```
  while <commands> do <commands> done
  until <commands> do <commands> done
  for <name> do <commands> done
  for <name> in <argument_list> do <commands> done
  if <commands> then <commands>
    [ elif <commands> then <commands> ]
    [ else <commands> ] fi
  case <name> in
    ( <muster>) <commands> ;; )
    esac
```

Trademarks: OS-9 (microware), UNIX (AT&T).
MSB-SH: Copyright (C) 1988 by MSB Software, Berlin

MSB micro computer system beratung
software · hardware · entwicklung
bernhard bardohl
[illegible] 28, tel. 030/624 78 61
1000 berlin 44

I B F
=====

ARK Corporation, Saitama, Japan, is pleased to announce the release of the IBF IEEE488/GP-IB File Manager for the OS-9/68K operating system.

IBF is a new file manager program that runs on MC680X0 based computers employing the OS-9/68K operating system. The file manager controls the IEEE488/GP-IB bus, an industry standard interface bus primarily designed for interfacing between computers and measuring instruments. IBF covers a wide range of applications from a simple measurement system with a computer and DVM (digital voltmeter), up to a complicated LAN (local area network) system.

Since IBF is not just a library package but an OS-9 file manager program, it works as a part of the operating system. IBF provides with a variety of IEEE488/GP-IB specific functions such as serial poll, parallel poll, pass control, and so on by system calls, as well as the common read/write entries that cooperate with Shell's redirection mechanism. For example, an IEEE488/GP-IB printer named "epson" simply prints the contents of a file by ente ing "list file >/epson" at Shell's prompt.

IBF transfers data by blocks for less overhead compared to SCF, which does by bytes. An IBF device driver can use DMA (direct memory access) for much faster transfer. IBF is suitable for applications requiring fast transfer speed, such as sweep measurements, light disks, image processing, and so on.

A nice feature of IBF is that it allows the user to register signal codes sent to the process when specific events occur. The events include data ready, SRQ, talker addressed, and so on. This synchronization mechanism with signals results efficient use of CPU time in OS-9's multiuser, multitasking environment.

The IBF Programming Package includes a set of library routines for application programs written in C. The library functions are compatible with the DIO library supported by the HP-UX (UNIX) operating system from Hewlett-Packard, the originator of the IEEE488/GP-IB standard. While IBF was designed aiming at strictly implementing the IEEE488 standard, it follows the ways of HP's desktop computers for several protocols left optional in the standard.

IBF is currently available for OEM licensing. Several Japanese computer manufacturers have signed contracts with ARK. The IBF Porting Package includes well-documented and portable sample device driver source code as well as a number of debugging utilities.

IBF SPECIFICATIONS

Target CPU: MC68000, MC68008, MC68010, MC68020 or MC68030
Operating System: OS-9/680X0 V2.2 or late
Recommended LSI: NEC uPD7210 or TI TMS9914A
Interface Functions: (an example with a uPD7210 LSI)

SH1	complete capability of source handshake interface
AH1	complete capability of acceptor handshake interface
T5	complete capability of talker interface
L3	complete capability of listener interface
SR1	complete capability of service request interface
RL0	no capability of remote/local interface
PP1,2	remote and local configuration of parallel poll interface
DC0	no capability of device clear interface
DT0	no capability of device trigger interface
C1	system controller interface capability
C2	send IFC, controller-in-charge capabilities
C3	send REN capability
C5	complete capability of controller

Transfer Modes: Text (I$Readln/I$Writln) and Binary (I$Read/I$Write)
DMA Transfer: supported by the device driver
Device Locking: lockable by a process

All inquiries contact:

Hiro Sugawara

ARK Corporation
Niizo 1021, Flower Heights #205
Toda-Shi, Saitama 335 JAPAN
PHONE:81-484-45-9020
FAX:81-484-45-9296

* IBF is a trademark of ARK Corporation. OS-9/68K is a trademark of Microware Systems Corporation. DIO and HP-UX is trademarks of Hewlett-Packard Company. UNIX is a trademark of AT&T.

FORCE COMPUTERS, INC
3165 Winchester Blvd
Campbell, CA 95008-6557

Corporate News

Prepared By:
Shohat & Davis PR
2959 S. Winchester Blvd, Campbell, CA 95008
Murry Shohat (408) 379-7434

Contacts:
FORCE: Wayne Fischer (408) 370-6300
Microware: Andy Crane (408) 980-0201

OS-9® Is Latest Multitasking Real-Time Operating System in the FORCE UNIX® Arsenal; Ports for 68000, 68020 and 68030 CPUs

CAMPBELL, CA., October 11, 1988 — OS-9/68000, one of the most popular real-time operating systems for computers based on the 680X0 microprocessor family, has been ported to VMEbus computers and related peripherals manufactured by FORCE COMPUTERS. OS-9/68000 and its companion resident and cross-development tools are produced by Microware Systems Corp. of Des Moines, Iowa.

"OS-9 is a modular, ROM-able operating system that meets the requirements of a wide spectrum of real-time applications from ROM-based controllers to networked disk-based systems," said Wayne Fischer, FORCE COMPUTERS' Director of Marketing. "Real-time customers using OS-9 can choose between resident application development or cross-development from UNIX or MS-DOS environments," he added.

The ports are part of a FORCE-Microware joint marketing agreement signed this fall. The agreement enables FORCE to sublicense OS-9 for the 16-bit CPU-6 and 32-bit CPU-29 and CPU-37 board-level computers. Customers who license Professional OS-9 are also entitled to 90-day support from Microware's technical hotline.

Intended for High Performance Multitasking Hardware

In addition to the CPU-6, CPU-29 and CPU-37, the ports include OS-9 drivers for several FORCE peripheral controllers. These include the FORCE WFC-1 (Winchester & Floppy controller), ISCSI-1 (intelligent SCSI-bus controller), ISIO-1 and SIO-2 (serial port controllers).

"Through our relationships with VMEbus board vendors, we have always sought to provide real-time system designers with outstanding hardware and software solutions. The power and flexibility of OS-9, coupled with FORCE's high performance hardware, greatly expands the options available to our customers," said Andy Bell, Vice President of Microware. "We're pleased that FORCE now provides OS-9 for use with their broad range of VMEbus products."

Two versions of OS-9 are available. The Professional OS-9 package provides an integrated disk-resident development environment and includes the OS-9 kernel, four file manager modules (pipe, serial, disk, tape), C Compiler, macro assembler/linker/symbolic debugger, uMACS screen editor, over 60 utilities and comprehensive documentation. The Industrial OS-9 package is intended for ROM-resident embedded applications and includes the OS-9 kernel, two file manager modules (pipe and serial), and a limited utility set.

For networking, FORCE customers can link OS-9 based systems to Ethernet networks using FORCE's ILANC-1 Ethernet controller combined with Microware's OS-9 Ethernet Support Package (ESP). ESP provides full remote login (TELNET), file transfer (FTP) and BSD 4.2 socket facilities between OS-9 and UNIX. Additional support for the CPU-37's on-board Ethernet capability will be available in the near future.

Compatible Development & Target Environments

Inexperienced OS-9 users are struck by the similarities to UNIX designed into the product's repertoire of features. OS-9 includes a similar file structure (including record locking), process model, shell user interface, socket facility and communications protocol (TCP/IP). These similarities, combined with C source code compatibility with UNIX, make OS-9 a powerful partner for cross development or distributed application systems.

However, OS-9 offers a choice of three host development environments. As a complete resident operating system, Professional OS-9 on FORCE hardware includes an excellent group of resident development tools. Microware's C Compiler is a comprehensive implementation of the Kernighan & Ritchie standard and supports fast IEEE "math" library and UNIX "cio" library, providing application portability. Professional OS-9 adds tools such as the Symbolic User State Debugger and OS-9 utility set, to minimize the application development cycle. Optional resident languages and tools include FORTRAN, Pascal, ADA and Basic, plus a C Source Level Debugger, networking and communications programs.

Customers who wish to use UNIX- or MS-DOS-based systems as the host have a variety of options. Microware's UNIX/OS-9 Cross C Compiler packages are available for a wide selection of UNIX systems, including SUN and MicroVAX systems. A new tool, OS-9 UniBridge, integrates a variety of modules for UNIX and OS-9 systems designed to support distributed C programming, remote debugging and UNIX supervision of real-time processes. "UniBridge will speed the recognition of UNIX as a powerful development environment for real-time process control," said Fischer.

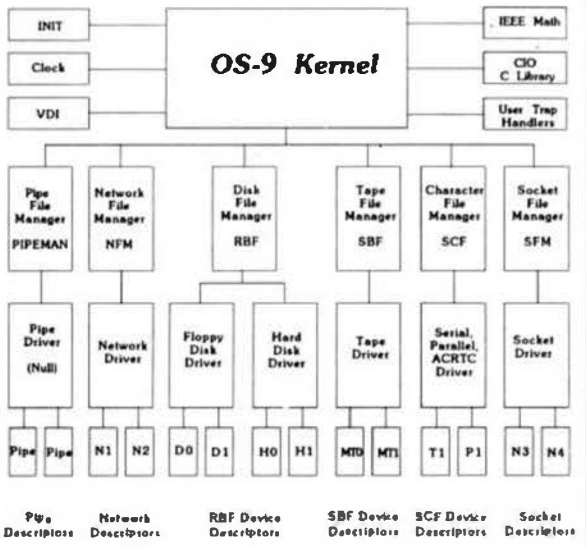
Another Microware tool, OS-9 PCBridge, turns an IBM PC or compatible system into a complete OS-9 C language development system. PCBridge features an MS-DOS/OS-9 Cross C Compiler, a Symbolic Debugger, plus terminal emulation, communication and file transfer utilities. When installed, PCBridge's menu-driven interface speeds the user through the process of generating application code for testing on the OS-9 target system.

Classifieds As Submitted - No Guarantees

30 Motorola MVME 133, CPU Module, 68020, 12.5 Mhz, 1 Meg DRAM, $675 ea.
30 Motorola MVME 225-1, 1 Meg Memory Module, $380 ea.
30 Motorola MVME 320A, Winchester/Floopy Controller, ST506 compatible, $490 ea.
30 Motorola MVME 332, Intelligent 8 Channel Serial Communications Module, 68010 supports multiple protocols, $675 ea.
35 Electronic Solutions 7 Slot VME Desktop Enclosures w/ 325 watt power supply, supports up to 4 mass storage devices, $695 ea.
2 Configured Systems. 40 Meg Seagate Winchester, 1 Meg Floopy, $3200 ea.
Call or write J.G. (602) 951-3373, RPO, POBox C6000, Suite 162, Scottsdale, Az. 85261 or
Tom Williams, (615) 842-4600.
MUSTANG-020 16Mhz with 68881. OS9 Professional Package & C $2500.
S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $2900.
HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.
3-Dual 8" drive enclosure with power supply. New in box. $125 each.
5-Siemens 8" Disk Drive, $100 each.
Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $250.
QUME QVT-102 terminal, like new, amber screen $250. or best offer.
SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card- $490 complete.
Tom (615) 842-4600 M-F 9AM to 5PM EST

U.S. Postal Service Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685): 1A. Title of Publication: 68 Micro Journal 1B. Publication no: 468510: 2. Date of Filing: 11-01-88: 3. Frequency of Issue: Monthly: 3A. No. of Issues Published> Annually: 12. 3B Annual Subscription Price: $24.50 4 and 5: Complete Mailing Address of Known Office, Headquarters or General Business Offices of the Publisher: 5900 Cassandra Smith Rd., Hixson, TN. 37343 6. Full Name and Complete Mailing Address of Publisher: Donald M. Williams Sr., 5900 Cassandra Smith Rd., Hixson, TN 37343. Editor: Donald M. Williams Sr., 5900 Cassandra Smith Rd., Hixson, TN 37343., Managing Editor: Larry E. Williams, 5900 Cassandra Smith Rd., Hixson, TN 37343 7.Owner: Computer Publishing Inc., 5900 Cassandra Smith Rd., Hixson, TN 37343, whose Stockholders are: Donald M. Williams, Sr., Frances J. Williams, Larry E. Williams, Mary E. Robertson, Thomas E. Williams. 8. Known Bondholders, Mortgagees, and other Security Holders Owning or Holding 1 Percent or more of Total Amount of Bonds, Mortgages or other Securities: None. 9. For Completion by Nonprofit Organizations Authorized to Mail at Special Rates: N/A 10. Extent and Nature of Circulation: A. Total No. Copies (Net Press Run): Average No. Copies Each Issue During Preceding 12 Months: 8110., Actual No. Copies of Single Issue Published Nearest to Filing Date: 7800. B. Paid and/or Requested Circulation: 1. Sales Through Dealers And Carriers,Street Vendors And Counter Sales: Average No. Copies Each Issue During Preceding 12 Months: 4890. Actual No. Copies Of Single Issue Published Nearest to Filing Date: 4710. 2. Mail Subscriptions: Average No. Copies Each Issue During Preceding 12 Months 2860. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 2749. C. Total Paid Circulation: Average No. Copies Each Issue During Preceding 12 Months: 7750. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 7459. D. Free Distribution By Mail, Carrier Or Other Means Including Samples, Complimentary, And Other Free Copies: Average No. Copies Each Issue During Preceding 12 Months: 54. Actual No. Copies Of Single Issue Published Nearest To Filing Date: 19. E. Total Distribution (Sum Of C And D): Average No. Copies Each Issue During Preceding 12 Months: 7804, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 7478. F. Copies Not Distributed: 1. Office Use, Left Over, Unaccounted, Spoiled After Printing: Average No. Copies Each Issue During Preceding 12 Months: 260, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 280. 2. Return From News Agents: Average No. Copies Each Issue During Preceding 12 Months: 46, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 42. G. Total:(Sum of E, F1 and 2-Should Equal Net Press Run Shown In A: Average No. Copies Each Issue During Preceding 12 Months: 8110, Actual No. Copies Of Single Issue Published Nearest To Filing Date: 7800. 11. I Certify That The Statements Made By Me Above Are Correct And Complete: (Signed): Larry E. Williams, Office Manager,68 Micro Journal. Computer Publishing, Inc. 12. For completion by publishers mailing at the regular rates: Permission requested.

OS-9 Module Organization



Modular Structure, Network Support

OS-9 is comprised of a set of independent modules dynamically linked by a memory management system at execution time. Each object in memory is named and kept in a stand ard format. A memory module directory containing information about each module is maintained by the OS-9 kernel. Multiple applications can dynamically link to memory modules, sharing common code and data to dramatically reduce system RAM requirements. Users can easily install or replace modules, including I/O drivers, while the system is running.

The nucleus of OS-9 is a full function multi-tasking kernel providing pre-emptive task scheduling, prioritized user-defined interrupt handling, dynamic memory management and over 70 system service facilities. The kernel can be used in a stand-alone environment or combined with optional hardware-independent I/O managers. This building block approach allows designers to easily tailor hardware and software modules to meet their applications requirements.

An important part of embedded control systems is the input/output capability. OS-9 provides I/O versatility and reliability while delivering outstanding performance. Along with independent I/O file manager modules for serial, disk, tape and socket support, OS-9 offers a hardware-independent Network File Manager. OS-9 Net links the I/O systems of OS-9 systems together transparently to provide full device sharing and interprocess communications facilities. OS-9 Net and Microware's ESP package can be combined to create a distributed development or application systems that build on the best features of both UNIX and OS-9.

Availability & Price

OS-9 is available immediately from FORCE for use with the CPU-6 and CPU-29 boards. Support for the CPU-37 will be released in 60 days. The CPU-6 is a 16-bit single board computer based on the 68000 microprocessor; the CPU-29 provides 32-bit performance based on the 68020 microprocessor; and the CPU-37 is a new high-performance single board computer based on the 32-bit 68030 microprocessor.

Professional OS-9 is priced at $1,250 for CPU-6, $1,500 for CPU-29 and $1,750 for CPU-37. This package includes drivers for other FORCE I/O boards. Industrial OS-9 is priced at $200, $300 and $400, respectively. The Ethernet Support Package (ESP) is priced at $1,100. PCBridge is priced at $1,400.

FOR THOSE WHO NEED TO KNOW

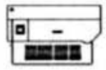
68 MICRO
JOURNAL™

VME Modules and Desk Top Enclosures:

Due to a cancelled project, Rumera General Partnership has a number of unused Motorola VME Modules and Electronic Solutions enclosures for sale at a substantial discount off list price.

Product	Part Number	Price
Motorola **68020** based CPU board w/ 1MB DRAM, 68881 FPP, 4 EPROM sockets, 3 serial ports and VMEbus interface.	MVME133	$675.00
Motorola 1 MB DRAM Module w/ A32/D32 VMEbus interface, byte parity and interleaving.	MVME225-1	$380.00
Motorola Winchester / Floppy Disk Controller. Supports 2 - 5.25 in. Wini's & 4 - 5.25 in. Floppies.	MVME320A	$490.00
Motorola 8 Channel Intelligent Communications Module w/ 68010 control & 128kb DRAM. Supports asynch RS232.	MVME332	$675.00
Electronic Solutions Desk Top Enclosure - 7 slot (Pl/P2) backplane, 325 W Power Supply, and mounting space for mass storage devices.	Series 7	$695.00

For information call or write to:
RGP
PO Box C6000, Ste 162
Scottsdale, AZ 85261
(602) 951-3373



ATARI & AMIGA CALL

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

This I must stress - *Input from you on the Atari & Amiga.* As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!

DMW

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDAROS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing **your OWN** more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
*** All Currency in U.S. Dollars**

Continually Updated In 68 Micro Journal Monthly

VISA

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615) 842-4601
Telex 5106006630

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware and Motorola
*68' Micro Journal is a trademark of Computer Publishing Inc.

!!! Subscribe Now !!!
68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard □ VISA □

Card #________________________ Exp. Date__________

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $____________

Name ______________________________

Street ______________________________

City __________________ State ____ Zip __________

Country ______________________________

My Computer Is: ______________

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

*Foreign Surface: Add $12.00 per Year to USA Price.

*Foreign Airmail: Add $48.00 per Year to USA Price.

*Canada & Mexico: Add $9.50 per Year to USA Price.

*U.S. Currency Cash or Check Drawn on a USA Bank !

68 Micro Journal
5900 Cassa dra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600
Telex 510 600-6630

Reader Service Disks

Disk- 1	Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2	Diskedit w/ inst.& fixes, Prime. *Prnod, **Snoopy, **Football, **Hexpawn,**Lifetime.
Disk- 3	Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp. *Disksave.
Disk- 4	Mailing Program, *Finddat, *Change, *Testdisk.
Disk- 5	*DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk-6	**Purchase Order, Index (Disk file indx).
Disk- 7	Linking Loader, Rload, Harkness.
Disk- 8	Crtest, Lanpher (May 82).
Disk- 9	Datacopy, Diskfix9 (Aug 82).
Disk-10	Home Accounting (July 82).
Disk-11	Dissembler (June 84).
Disk-12	Modem68 (May 84).
Disk-13	*Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt.
Disk-14	*Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib.
Disk-15	Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16	Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17	Match Utility, RATBAS, A Basic Preprocessor.
Disk-18	Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19	Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20	UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21	Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22	Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23	ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven, Language Recognition Utility, Anderson March 1986.
Disk-24	68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25	KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26	Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27	ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28	CT-82 Emulator, bit mapped.
Disk-29	**Star Trek
Disk-30	Simple Winchester, Dec.'86 Green.
Disk-31	*** Read/Write MS/PC-DOS (SK*DOS)
Disk-32	Heir-UNIX Type upgrade - 68MJ 2/87
Disk-33	Build the GT-4 Terminal - 68MJ 11/87 Condon.
Disk-34	FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88Korpi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

8" disk $19.50
5" disk $16.95

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

68 MICRO JOURNAL

5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

K-BASIC™

**The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems***

Hundreds Sold at Suggested Retail:

~~$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compiler their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug It in BASIC and Then Compile It to a .CMD Binary File.

For a LIMITED time save over 65%... This sale will not be repeated after it's over! *

SALE SPECIAL:

$69.95

SPECIAL

Thank-You-Sale

Only From:

S.E. Media™

CPI

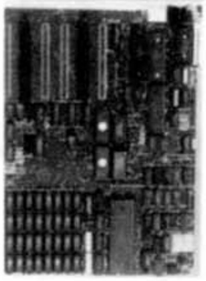
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems



BASIC KIT (8 MHZ) - Board, 68000, HUMBUG MONITOR + BASIC in ROM, 4K STATIC RAM, 2 SERIAL PORTS, all Components **$200**

PACKAGE DEAL - Complete Kit with Board 68000 10 MHZ, SK*DOS, 512K RAM, and all Necessary Parts **$575**

ASSEMBLED BOARD (12 MHZ)
Completely Tested, 1024K RAM, FLOPPY CONTROLLER, PIA, SK*DOS
$899

ASSEMBLED SYSTEM - 10 MHZ BOARD, CABINET POWER SUPPLY, MONITOR + KEYBOARD, 80 TRACK FLOPPY DRIVE, CABLES **$1299**
For A 20 MEG DRIVE, CONTROLLER and CABLES Add **$295**

PROFESSIONAL OS9 **$500**

FEATURES

- MC68000 Processor, 8 MHZ Clock (optional 10,12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K,64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

PERIPHERAL TECHNOLOGY

1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067
404/984-0742
VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue
For Complete Information On All Products

™SK*DOS is a Trademark of
STAR-K SOFTWARE SYSTEMS CORP.
™OS9 is a Trademark of Microware

DATA-COMP SPECIAL

Heavy Duty Power Supplies

For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

Make: Boschert
Size: 10.5 x 5 x 2.5 inches
Including heavy mounting bracket and heatsink

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery
SPECIAL: $59.95 each
2 or more $49.95 each
Add: $7.50 each S/H

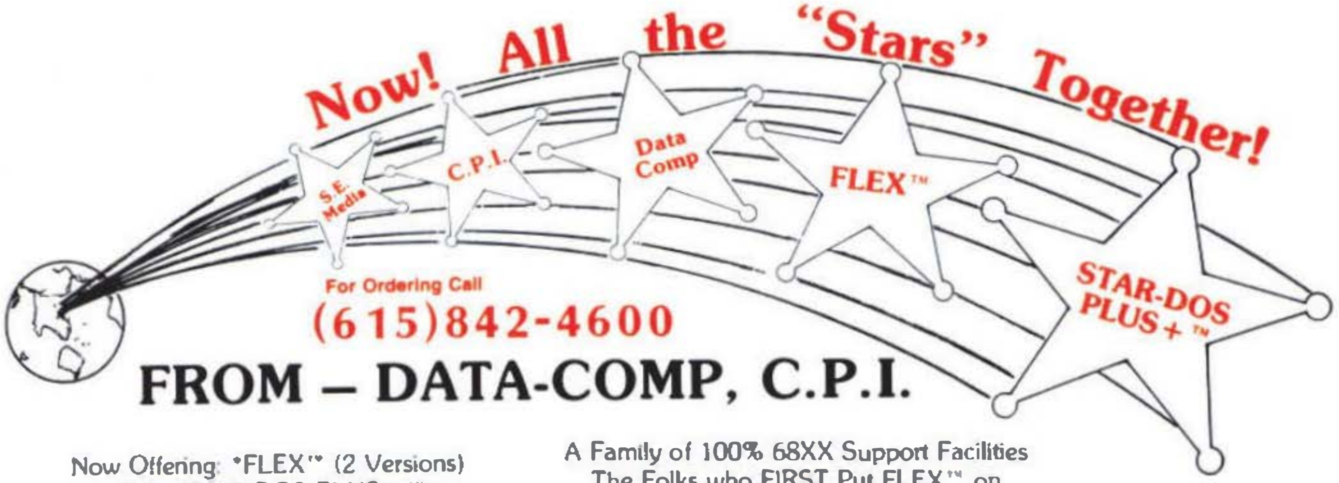
Make: Boschert
Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery
SPECIAL: $49.95 each
2 or more $39.95 each
Add: $7.50 S/H each

5900 Cassandra Smith Rd., Hixson, Tn. 37343 Telephone 615 842-4600 Telex 510 600-6630



FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+

- ● Functions Same as FLEX
- ● Reads - writes FLEX Disks
- ● Run FLEX Programs
- ● Just type: Run "STAR-DOS"
- ● Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

TSC Editor
Reg $50.00
NOW $35.00

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities

+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M NEW DISK CONTROLLER JPD-CP WITH J-DOS, RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Went J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JPD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95

CABLES & OTHER INTERFACES
CALL FOR PRICING

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

☎ (615)842-4600
For Ordering
Telex 5106006630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K

FULL 768K No Wait RAM

The **MUSTANG-08™** system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The **MUSTANG-08** includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed *whiz on disk I/O.* Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks.

SALE!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

*$400.00

See Mustang-02 Ad - page 5 for trade-in details

MUSTANG-08

LOOK

	Seconds	32 bit Integer	Register Long
Other 68008 8 Mhz OS-9	68K	18.0	9.0
MUSTANG-08 10 Mhz OS-9	68K	9.8	6.3

C Benchmark Loop

```
Main()
    {
        /* int I; */
        register long I;
        for (I=0; I < 999999; ++I);
    }
```

Now even faster! with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

25 Megabyte Hard Disk System

$2,398.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC

MUSTANG-08 is a trademark of CPI

Data-Comp Division

A Decade of Quality Service"

Systems World-Wide

Computer Publishing, Inc.
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4601 - Telex 510 600-6630

* Those with SWTPC hi-density FLEX 5" - Call for special info.